


```
RRRRRRRR      MM      MM      SSSSSSSS
RRRRRRRR      MM      MM      SSSSSSSS
RR      RR      MMMM      MMMM      SS
RR      RR      MMMM      MMMM      SS
RR      RR      MM      MM      SS
RR      RR      MM      MM      SS
RRRRRRRR      MM      MM      SSSSSS
RRRRRRRR      MM      MM      SSSSSS
RR      RR      MM      MM      SS
RR      RR      MM      MM      SS
RR      RR      MM      MM      SS
RR      RR      MM      MM      SS
RR      RR      MM      MM      SSSSSSSS
RR      RR      MM      MM      SSSSSSSS
```

```
....
....
....
....
```

```
LL      SSSSSSSS      TTTTTTTTTT
LL      SSSSSSSS      TTTTTTTTTT
LL      SS      TT
LL      SS      TT
LL      SS      TT
LL      SS      TT
LL      SSSSSS      TT
LL      SSSSSS      TT
LL      SS      TT
LL      SS      TT
LL      SS      TT
LL      SS      TT
LLLLLLLLLL      SSSSSSSS      TT
LLLLLLLLLL      SSSSSSSS      TT
```

```
0001 0  ++
0002 0
0003 0  UTLDEF.B32 - UTILITY DEFINITION MACROS FOR BLISS PROCESSING
0004 0  OF STARLET DEFINITION MACROS.
0005 0  Version 'V04-000'
0006 0
0007 0  --
0008 0  *****
0009 0  *
0010 0  * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0011 0  * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0012 0  * ALL RIGHTS RESERVED.
0013 0  *
0014 0  * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0015 0  * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0016 0  * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0017 0  * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0018 0  * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0019 0  * TRANSFERRED.
0020 0  *
0021 0  * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0022 0  * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0023 0  * CORPORATION.
0024 0  *
0025 0  * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0026 0  * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0027 0  *
0028 0  *
0029 0  *****
0030 0
0031 0  ++
0032 0
0033 0  MODIFIED BY:
0034 0
0035 0  V02-001 REFORMAT      Maria del C. Nasr      01-Aug-1980
0036 0
0037 0  --
0038 0
0039 0
0040 0  macros to extract offsets, field widths, etc., from field extraction macros
0041 0
0042 0
0043 0  MACRO  $BYTEOFFSET (OFFSET, POSITION, WIDTH, SIGN) = OFFSET%;
0044 0
0045 0  MACRO  $BITPOSITION (OFFSET, POSITION, WIDTH, SIGN) = POSITION%;
0046 0
0047 0  MACRO  $FIELDWIDTH (OFFSET, POSITION, WIDTH, SIGN) = WIDTH%;
0048 0
0049 0  MACRO  $EXTENSION (OFFSET, POSITION, WIDTH, SIGN) = SIGN%;
0050 0
0051 0  MACRO  $FIELDMASK (OFFSET, POSITION, WIDTH, SIGN) =
0052 0  (1^(POSITION+WIDTH) - 1^POSITION)%;
0053 0
0054 0  :
0055 0  : macro to generate eqlst constructs
0056 0
0057 0  MACRO
```

F 9
15-Sep-1984 22:56:00
15-Sep-1984 22:55:58

VAX-11 Bliss-32 V4.0-742
_S255SDUA28:[RMS.OBJ]RMS.R32;1

Page 2
(1)

.. M 0058 0
.. M 0059 0
.. M 0060 0
.. M 0061 0
.. M 0062 0
.. M 0063 0
.. M 0064 0
.. M 0065 0
.. M 0066 0
.. M 0067 0
.. M 0068 0
.. M 0069 0
.. 0070 0

```
$EQLST(P,G,I,S)[A]=
  %NAME(P,GET1ST_A) =
    %IF NUL2ND_A
      %THEN (I) + %COUNT*(S) ! assumes i, s always generated by conversion program
    %ELSE GET2ND_A
      %FI %
GET1ST_(A,B)=
  A-%
GET2ND_(A,B)=
  B-% ! known non-null
NUL2ND_(A,B)=
  %NULL(B) %;
```

\$BEGIN RMSIDXMAC,V04-000

MACRO DEFINITIONS FOR RMS-32 INDEX FILE ORGANIZATION

```
*****
*
*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
*  ALL RIGHTS RESERVED.
*
*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
*  ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
*  OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
*  TRANSFERRED.
*
*  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
*  AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
*  CORPORATION.
*
*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
*
*****
```

++

FACILITY: RMS32 INDEX SEQUENTIAL FILE ORGANIZATION

ABSTRACT:

ENVIRONMENT:

VAX/VMS OPERATING SYSTEM

--

AUTHOR: D. H. Gillespie CREATION DATE: 17-MAR-1978
 and W. Koenig

MODIFIED BY:

V03-002 MCN0003 Maria del C. Nasr 15-Mar-1982
 Use new general linkage for RM\$BUG3. Take out definition
 for R_REC_SIZE, and R_VBN.

V03-001 MCN0002 Maria del C. Nasr 25-Mar-1982
 Add macro definition to calculate key buffer address.

V02-003 CDS0001 C Saether 9-Dec-1981
 Comment out references to CSH\$M_READAHEAD flag in the
 \$CSHFLAGS macro.

: 0128 0
: 0129 0
: 0130 0
: 0131 0
: 0132 0
: 0133 0

:
:
: **

V02-002 MCN0001 Maria del C. Nasr 22-Apr-1981
Add macro definition for determining record identifier size

```

0134 0
0135 0
0136 0
0137 0
0138 0
0139 0
0140 0
0141 0
0142 0
0143 0
0144 0
0145 0
0146 0
0147 0
0148 0
0149 0
0150 0
0151 0
0152 0
0153 0
0154 0
0155 0
0156 0
0157 0
0158 0
0159 0
0160 0
0161 0
0162 0
0163 0
0164 0
0165 0
0166 0
0167 0
0168 0
0169 0
0170 0
0171 0
0172 0
0173 0
0174 0
0175 0
0176 0
0177 0
0178 0
0179 0
0180 0
0181 0
0182 0
0183 0
0184 0
0185 0
0186 0
0187 0
0188 0
0189 0
0190 0

: Define macro for psect attributes
MACRO PSECT_ATTR = EXECUTE,PIC,GLOBAL %;

: Define macro to extract size
MACRO $BYTESIZE(OFFSET,POSITION,WIDTH,SIGN) = WIDTH / 8 %;

: Structure declarations used for system defined structures to save typing
STRUCTURE
    BBLOCK [O, P, S, E; N] =
        [N]
        (BBLOCK+O)<P,S,E>,
    BBLOCKVECTOR [I, O, P, S, E; N, BS] =
        [N*BS]
        (BBLOCKVECTOR+(O+I*BS))<P,S,E>;

: ***** The following two macros violate the BLISS language definition
: ***** in that they make use of the value of SP while building the argument
: ***** list. It is the opinion of the BLISS maintainers that this usage is safe
: ***** from planned future optimizations.

: Macro to call the change mode to kernel system service.
: Macro call format is 'KERNEL_CALL( ROUTINE, ARG1, ARG2, ...).
MACRO
    KERNEL_CALL (R) =
        BEGIN
            EXTERNAL ROUTINE SY$CMKRNL : ADDRESSING_MODE (ABSOLUTE):
            BUILTIN SP;
            SY$CMKRNL(4, .SP, %LENGTH-1
                %IF %LENGTH GTR 1 %THEN, %REMAINING %FI)
        END%;

: macro to generate a string descriptor
MACRO
    DESCRIPTOR (STRING) =
        UPLIT (%CHARCOUNT (STRING), UPLIT BYTE (STRING))%;

: macro to return the number of actual parameters supplied to a routine call
MACRO
    ACTUALCOUNT =
        BEGIN
            BUILTIN AP;
            (.AP)<0,8>
        END
        %;
```

```

0191 0
0192 0
0193 0
0194 0
0195 0
0196 0
0197 0
0198 0
0199 0
0200 0
0201 0
0202 0
0203 0
0204 0
0205 0
0206 0
0207 0
0208 0
0209 0
0210 0
0211 0
0212 0
0213 0
0214 0
0215 0
0216 0
0217 0
0218 0
0219 0

! macro to generate call to bug check routine
MACRO BUG_CHECK =
  (LINKAGE L JSB;
   EXTERNAL ROUTINE
    RMSBUG3 : RLSJSB;
    RMSBUG3 ( ) %;

! Macro used to determine record identifier size (byte or word) depending on
! prologue version of the file.
MACRO IRC$ ID(RECADR) =
  (IF .IFAB[IFB$B_PLG_VER] LSSU PLG$C_VER_3
   THEN
    .RECADR[IRC$B_ID]
   ELSE
    .RECADR[IRC$W_ID]) % ;

! Macro used to determine address of key buffer wanted. Parameter is
! the keybuffer number.
MACRO KEYBUF_ADDR(KBUFNO) =
  .IRAB[IRB$L_KEYBUF] + .IFAB[IFB$W_KBUFSZ] * ((KBUFNO) - 1) % ;

```


!KEEP THESE DEFINITIONS IN ALPHABETICAL ORDER, PLEASE.

internal register definitions

common register definitions

MACRO

COMMON_FABREG =
R_FAB, R_IFAB, R_IFAB_FILE, R_IMPURE %.

COMMON_IOREG =
R_BDB, R_BKT_ADDR %.

COMMON_RABREG =
R_RAB, R_IRAB, R_IFAB, R_IMPURE%,

COMMON_FAB_STR =
R_FAB_STR,
R_IFAB_STR,
R_IFAB_FILE_STR,
R_IMPURE_STR %.

COMMON_IO_STR =
R_BDB_STR,
R_BKT_ADDR_STR %.

COMMON_RAB_STR =
R_RAB_STR,
R_IRAB_STR,
R_IFAB_STR,
R_IMPURE_STR%;

miscellaneous register definitions
the macros associated with registers will follow these conventions:
macro r_name =
name = register_number %; (no trailing punctuation)
to be used basically in a linkage statement, and
macro r_name_str =
r_name : ref bblock %; (or whatever structure)
to be used basically in external or global register declarations

MACRO

R_BDB =
BDB = 4 %.

R_BKT_ADDR =
BKT_ADDR = 5 %.

R_FAB =
FAB = 8 %.

R_IDX_DFN =
IDX_DFN = 7 %.

R_IFAB_FILE =
IFAB_FILE = 9 %.

R_IMPURE =
IMPURE = 11 %.

R_IRAB =

L 9
15-Sep-1984 22:56:00
15-Sep-1984 22:55:58

VAX-11 Bliss-32 V4.0-742
_S255\$DUA28:[RMS.OBJ]RMS.R32;1

Page 8
(4)

0277 0
M 0278 0
0279 0
M 0280 0
0281 0
M 0282 0
0283 0
0284 0
0285 0
M 0286 0
0287 0
M 0288 0
0289 0
M 0290 0
0291 0
M 0292 0
0293 0
M 0294 0
0295 0
M 0296 0
0297 0
M 0298 0
0299 0
M 0300 0
0301 0
M 0302 0
0303 0
M 0304 0
0305 0
M 0306 0
0307 0
M 0308 0
0309 0
0310 0

R_RAB = IRAB = 9 %,
R_RAB = RAB = 8 %,
R_REC_ADDR =
REC_ADDR = 6 %,
R_IFAB =
IFAB = 10 %,

R_BDB_STR =
R_BDB : REF BBLOCK %,
R_BKT_ADDR_STR =
R_BKT_ADDR : REF BBLOCK %,
R_FAB_STR =
R_FAB : REF BBLOCK %,
R_ID_STR =
R_ID : BYTE %,
R_IDX_DFN_STR =
R_IDX_DFN : REF BBLOCK %,
R_IFAB_STR =
R_IFAB : REF BBLOCK %,
R_IMPURE_STR =
R_IMPURE : REF BBLOCK %,
R_IRAB_STR =
R_IRAB : REF BBLOCK %,
R_RAB_STR =
R_RAB : REF BBLOCK %,
R_REC_ADDR_STR =
R_REC_ADDR : REF BBLOCK %,
R_IFAB_FILE_STR =
R_IFAB_FILE : REF BBLOCK %,
R_VBN_STR =
R_VBN %;

```

macro to make the status codes small
MACRO WORDMASK(CODE) = (
    (CODE AND %X'FFFF')
    %;
MACRO RMSEERR (CODE) = (
    %NAME('RMS$',CODE) AND %X'FFFF'
    %;
    RMSSUC (CODE) = (
        %IF %LENGTH EQL 0 %THEN
            1
        %ELSE
            %NAME('RMS$',CODE) AND %X'FFFF'
        %FI)
    %;
macro to make constants that are calculated have the <0,16> attribute

macro to make the code a little nicer
MACRO
this macro allows you to make a call to another routine
(and do whatever you want in a block before the call),
and if an error resulted, do whatever you want and
then return with the status.
RETURN ON ERROR (CALL) [] =
    (LOCAL STATUS;
    IF NOT (STATUS = (CALL)) THEN
        (%REMAINING;
        RETURN .STATUS)) %;

this macro is the same as the one above, except that it
it returns w/ status whether or not there was an error
and the caller can supply an 'else' block
note: the 'call' part and 'else' part must be separated by a comma
not a semicolon and the 'else' part must be terminated by a semicolon
RETURN_ELSE (CALL) [] =
    (LOCAL STATUS;
    IF NOT (STATUS = (CALL)) THEN RETURN .STATUS
    ELSE
        %REMAINING
        RETURN .STATUS)
    %;

```

```

! this is an internal cache macro to put the value of the flags into cshtmp
MACRO IRP(A)[] =
    %ASSIGN (CSHTMP,CSHTMP OR %NAME('CSHSM_',A));
    IRP(%REMAINING);
    %;

! this is an internal cache macro to verify the cache flags and set them up
MACRO %CSHFLAGS(FLAGS) =
    COMPILETIME CSHTMP = 0;
    %IF NOT %NULL(FLAGS) %THEN
        IRP (%REMOVE(FLAGS));
    %FI;
    %IF (CSHTMP AND CSHSM_READAHEAD) NEQ 0 %THEN
        %ASSIGN (CSHTMP,CSHTMP OR CSHSM_NOWAIT);
        %IF (CSHTMP AND
            (CSHSM_LOCK OR CSHSM_NOREAD OR CSHSM_NOBUFFER))
            NEQ 0 %THEN
            %ERRORMACRO ('INVALID CACHE FLAG COMBINATION');
        %FI;
    %FI;
    %IF (CSHTMP AND CSHSM_NOBUFFER) NEQ 0 %THEN
        %ASSIGN (CSHTMP,CSHTMP OR CSHSM_NOREAD);
    %FI;
    %;

! this is a macro to call cache or cachec
MACRO CACHE (VBN,SIZE,FLAGS,EP) =
    BEGIN
        %IF %NULL(FLAGS) %THEN
            COMPILETIME CSHTMP = 0
        %ELSE
            %CSHFLAGS(FLAGS)
        %FI;
        %IF %NULL(EP) %THEN
            RM$CACHE(VBN,SIZE,CSHTMP)
        %ELSE
            %NAME('RM$CACHE',EP)(VBN,SIZE,CSHTMP)
        %FI
    END
    %;

! this is a macro to call getbkt or getbktc
MACRO GETBKT (VBN,SIZE,FLAGS,EP) =
    BEGIN
        %IF %NULL(FLAGS) %THEN
            COMPILETIME CSHTMP = 0
        %ELSE
            %CSHFLAGS(FLAGS)
        %FI;
        %IF %NULL(EP) %THEN
            RM$GETBKT(VBN,SIZE,CSHTMP)
        %ELSE
            %NAME('RM$GETBKT',EP)(VBN,SIZE,CSHTMP)
        %FI
    END

```

B 10
15-Sep-1984 22:56:00
15-Sep-1984 22:55:58

VAX-11 Bliss-32 V4.0-742
_S255SDUA28:[RMS.OBJ]RMS.R32;1

Page 11
(6)

: 0418 0
: 0419 0
: 0420 0
: 0421 0

%;

```
0422 0
0423 0
0424 0
0425 0
0426 0
0427 0
0428 0
0429 0
0430 0
0431 0
0432 0
0433 0
0434 0
0435 0
0436 0
0437 0
0438 0
0439 0
0440 0
0441 0
0442 0
0443 0
0444 0
0445 0
0446 0
0447 0
0448 0
0449 0
0450 0
0451 0
0452 0
0453 0
0454 0
0455 0
0456 0
0457 0
0458 0

! these are macros to do probing of user structures
MACRO
  IFNORD (SIZE,ADDR,MODE) [] =
  (
    IF NOT PROBER(
      %IF %NULL(MODE) %THEN 0 %ELSE MODE %FI,
      SIZE,
      ADDR)
    THEN
      %REMAINING)
  %,
  IFNOWRT (SIZE,ADDR,MODE) [] =
  (
    IF NOT PROBEW(
      %IF %NULL(MODE) %THEN 0 %ELSE MODE %FI,
      SIZE,
      ADDR)
    THEN
      %REMAINING)
  %,
  IFRD (SIZE,ADDR,MODE) [] =
  (
    IF PROBER(
      %IF %NULL(MODE) %THEN 0 %ELSE MODE %FI,
      SIZE,
      ADDR)
    THEN
      %REMAINING)
  %,
  ! macros to do long probes
  READ_LONG(SIZE,ADDR,MODE) = NOT RMS$NOREAD_LONG(SIZE,ADDR,MODE) %,
  WRT_LONG(SIZE,ADDR,MODE) = NOT RMS$NOWRT_LONG(SIZE,ADDR,MODE) %;
```

```
0459 0
0460 0 macro to release a bucket and clear the location where its bdb addr is stored
0461 0
M 0462 0 MACRO RELEASE(B) =
M 0463 0 BEGIN
M 0464 0 BDB = .B;
M 0465 0 B = 0;
M 0466 0 RMSRL$BKT(0);
0467 0 END%;
0468 0
0469 0 macro to make sure that an assumption made about the position of symbols
0470 0 in a structure is still valid
0471 0 the arguments to this macro must be preceded by %quote
0472 0 e.g., assume (%quote ifb$b_bid, %quote ifb$b_bln);
0473 0
M 0474 0 MACRO ASSUME (A,B) =
M 0475 0 %IF $BYTEOFFSET(A) + $BYTESIZE(A) NEQ $BYTEOFFSET(B)
M 0476 0 %THEN %WARN('WARNING CONSTANT HAS CHANGED')
0477 0 %FI %;
0478 0
0479 0 this version of assume is good for constants
0480 0 e.g. assume (irc$c_fixovhdsz + 2, irc$c_varovhdsz);
0481 0
M 0482 0 MACRO ASSUME C (A,B) =
M 0483 0 %IF $BYTEOFFSET(A) NEQ $BYTEOFFSET(B)
M 0484 0 %THEN %WARN('WARNING CONSTANT HAS CHANGED')
0485 0 %FI %;
0486 0 [ 2 0 1 , 1 0 ] R M S I D X L N K . R 3 2
0487 0
0488 0 Define subroutine linkage
0489 0
0490 0
0491 0 *****
0492 0 *
0493 0 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0494 0 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0495 0 * ALL RIGHTS RESERVED.
0496 0 *
0497 0 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0498 0 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0499 0 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0500 0 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0501 0 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0502 0 * TRANSFERRED.
0503 0 *
0504 0 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0505 0 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0506 0 * CORPORATION.
0507 0 *
0508 0 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0509 0 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0510 0 *
0511 0 *
0512 0 *****
0513 0
0514 0 ++
0515 0
```

0516 0
0517 0
0518 0
0519 0
0520 0
0521 0
0522 0
0523 0
0524 0
0525 0
0526 0
0527 0
0528 0
0529 0
0530 0
0531 0
0532 0
0533 0
0534 0
0535 0
0536 0
0537 0
0538 0
0539 0
0540 0
0541 0
0542 0
0543 0
0544 0
0545 0
0546 0
0547 0
0548 0
0549 0
0550 0
0551 0
0552 0
0553 0

FACILITY: RMS32 INDEX SEQUENTIAL FILE ORGANIZATION

ABSTRACT: This module defines all the routine linkage

ENVIRONMENT:
VAX/VMS OPERATING SYSTEM

--
AUTHOR: D. H. Gillespie CREATION DATE: 17-MAR-1978
and W. Koenig

MODIFIED BY:

V03-024 RAS0154 Ron Schaefer 2-May-1983
Add NOPRESERVE (R2) to L_EXTEND0 linkage.
V03-023 MCN0020 Maria del C. Nasr 07-Apr-1983
Eliminate linkages of RMSNULLKEY, and RMSCOMPRESS KEY.
They will be using general linkages. Modify L_ACLOC3,
and L_EXTEND0 to use parameters instead of global registers.
V03-022 MCN0019 Maria del C. Nasr 05-Apr-1983
Preserve all registers except R0 and R1 in linkage
FABREG. RMSXSUM0 requires a separate linkage because
it cannot preserve R4.
V03-021 TMK0001 Todd M. Katz 26-Mar-1983
Add the linkage RABREG_4.
V03-020 MCN0018 Maria del C. Nasr 24-Mar-1983
Define new general linkages. Also, since the linkages
have changed so much, eliminate all history comments.

This module defines all the routine linkage for RMS-32 index file organization.

KEEP THESE DEFINITIONS IN ALPHABETICAL ORDER PLEASE

The following conventions will be used for linkage macros:

```
MACRO L_NAME =
    RL$NAME =
    JSB (REGISTERS) :
    GLOBAL (REGISTER DEFINITIONS) %;
```

The register definitions are macros of the forms
COMMON FABREG, COMMON RABREG, COMMON IOREG, etc.
or R_REGNAME as described in RMSIDXMÄC.R32

MACRO

```
L_ALDBUF =
    RL$ALDBUF =
    JSB (REGISTER = 5) :
    GLOBAL (R_IMPURE, R_IFAB)
    NOPRESERVE (2,3,4)
    NOTUSED (8,9) %;
```

```
L_ALLOC3 =
    RL$ALLOC3 =
    JSB (REGISTER = 7; REGISTER = 1, REGISTER = 2) :
    GLOBAL (R_IFAB) %;
```

```
L_BDBALLOC =
    RL$BDBALLOC =
    JSB (REGISTER = 4, REGISTER = 5) :
    GLOBAL (COMMON RABREG)
    NOPRESERVE (2,3,4,5,6) %;
```

```
L_CACHE =
    RL$CACHE =
    JSB (REGISTER = 1, REGISTER = 2, REGISTER = 3) :
    GLOBAL (COMMON IOREG)
    NOPRESERVE (1,2,3)
    NOTUSED (8,9,10,11) %;
```

```
L_CHECK_SEGMENT =
    RL$CHECK_SEGMENT =
    JSB (REGISTER = 0, REGISTER = 4, REGISTER = 2) :
    GLOBAL (R_IDX_DFN)
    NOPRESERVE (2,4,5)
    PRESERVE (1) %;
```

```
L_CHKSUM =
    RL$CHKSUM =
    JSB (REGISTER = 5) :
    NOPRESERVE (0,1,2) %;
```

```
L_COMPARE_KEY =
```

```
0554 0
0555 00
0556 00
0557 00
0558 00
0559 00
0560 00
0561 00
0562 00
0563 00
0564 00
0565 00
0566 00
0567 00
0568 00
0569 00
0570 00
0571 00
0572 00
0573 00
0574 00
0575 00
0576 00
0577 00
0578 00
0579 00
0580 00
0581 00
0582 00
0583 00
0584 00
0585 00
0586 00
0587 00
0588 00
0589 00
0590 00
0591 00
0592 00
0593 00
0594 00
0595 00
0596 00
0597 00
0598 00
0599 00
0600 00
0601 00
0602 00
0603 00
0604 00
0605 00
0606 00
0607 00
0608 00
0609 00
0610 0
```

```

0611 0
0612 0
0613 0
0614 0
0615 0
0616 0
0617 0
0618 0
0619 0
0620 0
0621 0
0622 0
0623 0
0624 0
0625 0
0626 0
0627 0
0628 0
0629 0
0630 0
0631 0
0632 0
0633 0
0634 0
0635 0
0636 0
0637 0
0638 0
0639 0
0640 0
0641 0
0642 0
0643 0
0644 0
0645 0
0646 0
0647 0
0648 0
0649 0
0650 0
0651 0
0652 0
0653 0
0654 0
0655 0
0656 0
0657 0
0658 0
0659 0
0660 0
0661 0
0662 0
0663 0
0664 0
0665 0
0666 0
0667 0

RL$COMPARE_KEY =
JSB (REGISTER = 1, REGISTER = 3, REGISTER = 0) :
GLOBAL (R_IDX_DFN)
NOPRESERVE (3) %,

L_ERROR_LINK1 =
RL$ERROR_LINK1 =
JSB () :
GLOBAL (COMMON_RABREG)
PRESERVE (0) %,

L_ERROR_LINK2 =
RL$ERROR_LINK2 =
JSB () :
GLOBAL (COMMON_RABREG, R_IDX_DFN)
PRESERVE (0) %,

L_EXTENDO =
RL$EXTENDO =
JSB (REGISTER = 5, REGISTER = 6; REGISTER = 1, REGISTER = 6) :
GLOBAL (COMMON_FABREG)
NOPRESERVE (2,3,4,5) %,

L_FABREG =
RL$FABREG =
JSB () :
GLOBAL (COMMON_FABREG)
NOPRESERVE (0,T) %,

L_FABREG_7 =
RL$FABREG_7 =
JSB () :
GLOBAL (COMMON_FABREG, R_IDX_DFN) %,

L_GETSPC =
RL$GETSPC =
JSB (REGISTER = 1, REGISTER = 2; REGISTER = 1) :
GLOBAL (R_IMPURE)
NOPRESERVE (2,3,4)
NOTUSED (8,9,10) %,

L_JSB =
RL$JSB =
JSB () %,

L_JSB01 =
RL$JSB01 =
JSB (REGISTER = 0, REGISTER = 1) :
GLOBAL (R_BKT_ADDR, R_REC_ADDR, R_IDX_DFN, R_IRAB, R_IFAB)
NOPRESERVE (0,1) %,

L_LINK_7_10_11 =
RL$LINK_7_10_11 =
JSB () :
GLOBAL (R_IDX_DFN, R_IFAB, R_IMPURE)
NOPRESERVE (0,1) %,

```

H 10
15-Sep-1984 22:56:00
15-Sep-1984 22:55:58

VAX-11 Bliss-32 V4.0-742
_S255\$DUA28:[RMS.OBJ]RMS.R32;1

Page 17
(9)

```
L_PRESERVE1 =
    RL$PRESERVE1 =
    JSB () :
    GLOBAL (COMMON_RABREG, R_BDB, R_REC_ADDR, R_IDX_DFN)
    PRESERVE (1) %,

L_QUERY_AND_LOCK =
    RL$QUERY_AND_LOCK =
    JSB (REGISTER = 1, REGISTER = 2) :
    GLOBAL (COMMON_RABREG)
    NOPRESERVE (3) %,

L_RABREG =
    RL$RABREG =
    JSB () :
    GLOBAL (COMMON_RABREG)
    NOPRESERVE (0,T) %,

L_RABREG_4 =
    RL$RABREG_4 =
    JSB () :
    GLOBAL (COMMON_RABREG, R_BDB)
    NOPRESERVE (0,T) %,

L_RABREG_4567 =
    RL$RABREG_4567 =
    JSB () :
    GLOBAL (COMMON_RABREG, COMMON_IOREG, R_REC_ADDR, R_IDX_DFN)
    NOPRESERVE (0,T) %,

L_RABREG_457 =
    RL$RABREG_457 =
    JSB () :
    GLOBAL (COMMON_RABREG, COMMON_IOREG, R_IDX_DFN)
    NOPRESERVE (0,T) %,

L_RABREG_467 =
    RL$RABREG_467 =
    JSB () :
    GLOBAL (COMMON_RABREG, R_BDB, R_REC_ADDR, R_IDX_DFN)
    NOPRESERVE (0,T) %,

L_RABREG_567 =
    RL$RABREG_567 =
    JSB () :
    GLOBAL (COMMON_RABREG, R_BKT_ADDR, R_REC_ADDR, R_IDX_DFN)
    NOPRESERVE (0,T) %,

L_RABREG_67 =
    RL$RABREG_67 =
    JSB () :
    GLOBAL (COMMON_RABREG, R_REC_ADDR, R_IDX_DFN)
    NOPRESERVE (0,T) %,

L_RABREG_7 =
    RL$RABREG_7 =
    JSB () :
```

I 10
15-Sep-1984 22:56:00
15-Sep-1984 22:55:58

VAX-11 Bliss-32 V4.0-742
_5255\$DUA28:[RMS.OBJ]RMS.R32;1

Page 18
(9)

0725 0
0726 0
0727 0
0728 0
0729 0
0730 0
0731 0
0732 0
0733 0
0734 0
0735 0
0736 0
0737 0
0738 0
0739 0
0740 0
0741 0
0742 0
0743 0
0744 0
0745 0
0746 0
0747 0
0748 0
0749 0
0750 0
0751 0
0752 0
0753 0
0754 0
0755 0
0756 0
0757 0
0758 0
0759 0
0760 0
0761 0
0762 0
0763 0
0764 0
0765 0
0766 0
0767 0
0768 0
0769 0
0770 0
0771 0
0772 0
0773 0
0774 0
0775 0
0776 0
0777 0
0778 0
0779 0
0780 0
0781 0

GLOBAL (COMMON RABREG, R_IDX_DFN)
NOPRESERVE (0,T) %;

L_REC_OVHD =
RL\$REC OVHD =
JSB (REGISTER = 1; REGISTER = 1) :
GLOBAL (R_REC_ADDR, R_IDX_DFN, R_IFAB) %;

L_RELEASE =
RL\$RELEASE =
JSB (REGISTER = 3) :
GLOBAL (R_BDB, R_IRAB, R_IFAB, R_IMPURE)
NOPRESERVE (1,2)
NOTUSED (8) %;

L_RELEASE_FAB =
RL\$RELEASE_FAB =
JSB (REGISTER = 3) :
GLOBAL (R_BDB, R_IFAB, R_IFAB_FILE, R_IMPURE)
NOPRESERVE (1,2)
NOTUSED(8) %;

L_RETSPC =
RL\$RETSPC =
JSB (REGISTER= 2, REGISTER = 3, REGISTER = 4) :
GLOBAL (R_IMPURE)
NOPRESERVE (2,3,5)
NOTUSED (8,9,10) %;

L_SIDR_FIRST =
RL\$SIDR_FIRST =
JSB (STANDARD; REGISTER = 1, REGISTER = 2) :
GLOBAL (R_REC_ADDR, R_IDX_DFN, COMMON_RABREG) %;

L_XSUMO =
RL\$XSUMO =
JSB () :
GLOBAL (COMMON FABREG)
NOPRESERVE (0,T,4) %;

VERSION: 'V04-000'

*
* COPYRIGHT (C) 1978, 1980, 1982, 1984 BY
* DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
* ALL RIGHTS RESERVED.
*
*

*
* THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
* ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
* INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
* COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
* OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
* TRANSFERRED.
*

```
* THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE *
* AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT *
* CORPORATION. *
*
* DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS *
* SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL. *
*
*****
**
FACILITY:

    MESSAGES ARE FOR THE FAL (DECNET FILE ACCESS LISTENER) FACILITY, BUT
    ARE GENERATED ON BEHALF OF FAL BY RMS.

ABSTRACT:

    THIS MODULE DEFINES FAL STATUS CODE SYMBOLS AND CORRESPONDING MESSAGE
    TEXT. NOTE THAT THIS MESSAGE FILE IS MAINTAINED BY THE RMS FACILITY,
    NOT THE FAL FACILITY, BECAUSE RMS DOES THE TRANSLATION OF DAP STATUS
    INTO FAL STATUS CODES WHICH ARE REPORTED AS SECONDARY STATUS INFORMATION
    TO THE RMSS_NETFAIL AND RMSS_SUPPORT COMPLETION CODES ON BEHALF OF FAL.

ENVIRONMENT: VAX/VMS

AUTHOR: JAMES A. KRYCKA,      CREATION DATE: 15-JAN-1982

MODIFIED BY:

    V03-001 JAK0147      J A KRYCKA      09-JUL-1984
    ADD ERROR CODES THAT CORRESPOND TO DAP MICCODES 416 TO 470
    (OCTAL) FOR MACCODES 4 THROUGH 7 DEFINED IN THE DAP V7.0
    SPECIFICATION.

--
**
DEFINE FAL STATUS CODES FOR USE BY RMS AS SECONDARY STATUS CODES.

LAYOUT OF MESSAGE SPACE:

    MSG NUMBERS      MSG NUMBERS      MSG CODES      DESCRIPTION
    (DECIMAL)        (HEXADECIMAL)    (HEXADECIMAL)
    0 - 1023         0000 - 03FF      8000 - 9FFF      UNUSED
    1024 - 2047      0400 - 07FF      A000 - BFFF      FOR RMSS_NETFAIL
    2048 - 4095      0800 - 0FFF      C000 - FFFF      FOR RMSS_SUPPORT

--
!...$RMSFALMSG

LITERAL
$EQUISH (FALS,GBL,0,1
    ,(FACILITY,          503)

**
DEFINE FAL STATUS CODES THAT ARE ASSOCIATED WITH THE RMSS_NETFAIL COMPLETION
CODE AND RETURNED IN THE STV FIELD OF THE FAB OR RAB.
```

P 0839 0
 P 0840 0
 P 0841 0
 P 0842 0
 P 0843 0
 P 0844 0
 P 0845 0
 P 0846 0
 P 0847 0
 P 0848 0
 P 0849 0
 P 0850 0
 P 0851 0
 P 0852 0
 P 0853 0
 P 0854 0
 P 0855 0
 P 0856 0
 P 0857 0
 P 0858 0
 P 0859 0
 P 0860 0
 P 0861 0
 P 0862 0
 P 0863 0
 P 0864 0
 P 0865 0
 P 0866 0
 P 0867 0
 P 0868 0
 P 0869 0
 P 0870 0
 P 0871 0
 P 0872 0
 P 0873 0
 P 0874 0
 P 0875 0
 P 0876 0
 P 0877 0
 P 0878 0
 P 0879 0
 P 0880 0
 P 0881 0
 P 0882 0
 P 0883 0
 P 0884 0
 P 0885 0
 P 0886 0
 P 0887 0
 P 0888 0
 P 0889 0
 P 0890 0
 P 0891 0
 P 0892 0
 P 0893 0
 P 0894 0
 P 0895 0

USE THE FOLLOWING FORMULA TO MAP A DAP STATUS CODE (STSCODE FIELD WITH
 MACCODE = 4 THRU 7) INTO A CORRESPONDING FAL MESSAGE CODE (LOWER 16 BITS):

$$\text{MESSAGE-CODE} = (2 \times 15 + ((\text{MICCODE} + 1024) \times 8) + \text{SEVERITY-LEVEL})$$

NOTE: THE FIRST THREE CHARACTERS OF THE MESSAGE IDENTIFICATION MNEMONIC
 FOR EACH MESSAGE CODE IN THIS SET INDICATES ITS ORIGIN AS FOLLOWS:

RMS --- THE CODE IS DEFINED FOR RMS-11 OR RMS-20, BUT NOT FOR RMS-32.
 (RMS-32 COMPLETION CODES ARE DEFINED IN RMSDEF.MDL.)
 FCS --- THE CODE IS DEFINED FOR THE FCS-11 ENVIRONMENT.
 TEN --- THE CODE IS DEFINED FOR THE TOPS-10 ENVIRONMENT.
 DAP --- ORIGIN IN THE DAP SPECIFICATION IS UNKNOWN.

	(OFFSET A,	1024)
! MICCODE = 0000 (OCTAL)	(DAPFAIL,	(%X'01F7A004'))
	(RMSABO,	(%X'01F7A00C'))
! MICCODE = 0011 (OCTAL)	(RMSAST,	(%X'01F7A04C'))
	(RMSBPA,	(%X'01F7A054'))
	(RMSBPS,	(%X'01F7A05C'))
! MICCODE = 0026 (OCTAL)	(RMSCLS,	(%X'01F7A0B4'))
! MICCODE = 0056 (OCTAL)	(RMSFID,	(%X'01F7A174'))
! MICCODE = 0071 (OCTAL)	(RMSINI,	(%X'01F7A1CC'))
! MICCODE = 0102 (OCTAL)	(RMSLBL,	(%X'01F7A214'))
	(RMSLBY,	(%X'01F7A21C'))
	(RMSLCH,	(%X'01F7A224'))
! MICCODE = 0106 (OCTAL)	(RMSLOC,	(%X'01F7A234'))
	(RMSMAP,	(%X'01F7A23C'))
! MICCODE = 0115 (OCTAL)	(RMSNID,	(%X'01F7A26C'))
! MICCODE = 0117 (OCTAL)	(RMSOPN,	(%X'01F7A27C'))
! MICCODE = 0124 (OCTAL)	(RMSPRM,	(%X'01F7A2A4'))
! MICCODE = 0153 (OCTAL)	(RMSSTK,	(%X'01F7A35C'))
! MICCODE = 0162 (OCTAL)	(RMSVOL,	(%X'01F7A394'))
! MICCODE = 0170 (OCTAL)	(RMSCAA,	(%X'01F7A3C4'))
! MICCODE = 0231 (OCTAL)	(RMSBLK,	(%X'01F7A4CC'))
	(RMSBSZ,	(%X'01F7A4D4'))
	(RMSCDR,	(%X'01F7A4DC'))
	(RMSCGJ,	(%X'01F7A4E4'))
	(RMSCOF,	(%X'01F7A4EC'))
	(RMSJFN,	(%X'01F7A4F4'))
	(RMSPEF,	(%X'01F7A4FC'))
	(RMSTRU,	(%X'01F7A504'))

L 10
15-Sep-1984 22:56:00
15-Sep-1984 22:55:58

VAX-11 Bliss-32 V4.0-742
_ \$255\$DUA28:[RMS.OBJ]RMS.R32;1

Page 21
(9)

```

P 0896 0      .(RMSUDF,      (XX'01F7A50C'))
P 0897 0      .(RMSXCL,      (XX'01F7A514'))
P 0898 0      .(DAPDIRFUL,    (XX'01F7A51C'))
P 0899 0      .(FCSHWR,      (XX'01F7A524'))
P 0900 0      .(FCSFHE,      (XX'01F7A52C'))
P 0901 0      .(DAPWRTEOF,    (XX'01F7A534'))
P 0902 0      .(FCSONP,      (XX'01F7A53C'))
P 0903 0      .(FCSDNA,      (XX'01F7A544'))
P 0904 0      .(FCSDA,      (XX'01F7A54C'))
P 0905 0      .(FCSUN,      (XX'01F7A554'))
P 0906 0      .(FCSRSU,      (XX'01F7A55C'))
P 0907 0      .(FCSOVR,      (XX'01F7A564'))
P 0908 0      .(FCSBCC,      (XX'01F7A56C'))
P 0909 0      .(FCSNOD,      (XX'01F7A574'))
P 0910 0      .(FCSIFU,      (XX'01F7A57C'))
P 0911 0      .(FCSHFU,      (XX'01F7A584'))
P 0912 0      .(FCSWAC,      (XX'01F7A58C'))
P 0913 0      .(FCSCKS,      (XX'01F7A594'))
P 0914 0      .(FCSWAT,      (XX'01F7A59C'))
P 0915 0      .(FCSALN,      (XX'01F7A5A4'))
P 0916 0      .(FCSBTF,      (XX'01F7A5AC'))
P 0917 0      .(FCSILL,      (XX'01F7A5B4'))
P 0918 0      .(FCS2DV,      (XX'01F7A5BC'))
P 0919 0      .(FCSFEX,      (XX'01F7A5C4'))
P 0920 0      .(FCSRNM,      (XX'01F7A5CC'))
P 0921 0      .(FCSFOP,      (XX'01F7A5D4'))
P 0922 0      .(FCSVER,      (XX'01F7A5DC'))
P 0923 0      .(FCSEOV,      (XX'01F7A5E4'))
P 0924 0      .(FCSDAO,      (XX'01F7A5EC'))
P 0925 0      .(FCSBBE,      (XX'01F7A5F4'))
P 0926 0      .(FCSEOT,      (XX'01F7A5FC'))
P 0927 0      .(FCSNBF,      (XX'01F7A604'))
P 0928 0      .(FCSNBK,      (XX'01F7A60C'))
P 0929 0      .(FCSNST,      (XX'01F7A614'))
P 0930 0      .(FCSULK,      (XX'01F7A61C'))
P 0931 0      .(FCSNLN,      (XX'01F7A624'))
P 0932 0      .(FCSSRE,      (XX'01F7A62C'))
P 0933 0      ! MICCODE = 0311 (OCTAL)
P 0934 0      .(DAPQUOEXC,    (XX'01F7A64C'))
P 0935 0      ! MICCODE = 0346 (OCTAL)
P 0936 0      .(DAPDIRCAF,    (XX'01F7A734'))
P 0937 0      .(DAPDIRCRA,    (XX'01F7A73C'))
P 0938 0      .(DAPDIRPRO,    (XX'01F7A744'))
P 0939 0      .(DAPDIRPRA,    (XX'01F7A74C'))
P 0940 0      .(DAPDIRNFA,    (XX'01F7A754'))
P 0941 0      .(DAPDIRCON,    (XX'01F7A75C'))
P 0942 0      ! MICCODE = 0416 (OCTAL)
P 0943 0      .(TENFILMOD,    (XX'01F7A874'))
P 0944 0      .(TENDEVNA,     (XX'01F7A87C'))
P 0945 0      .(TENDEVNF,     (XX'01F7A884'))
P 0946 0      .(TENPARALL,    (XX'01F7A88C'))
P 0947 0      .(TENBNFREE,    (XX'01F7A894'))
P 0948 0      .(TENCSD,      (XX'01F7A89C'))
P 0949 0      .(TENCDDF,      (XX'01F7A8A4'))
P 0950 0      .(TENSFDNF,     (XX'01F7A8AC'))
P 0951 0      .(TENSLE,      (XX'01F7A8B4'))
P 0952 0      .(TENSFDCNT,    (XX'01F7A8BC'))
```

P 0953 0
P 0954 0
P 0955 0
P 0956 0
P 0957 0
P 0958 0
P 0959 0
P 0960 0
P 0961 0
P 0962 0
P 0963 0
P 0964 0
P 0965 0
P 0966 0
P 0967 0
P 0968 0
P 0969 0
P 0970 0
P 0971 0
P 0972 0
P 0973 0
P 0974 0
P 0975 0
P 0976 0
P 0977 0
P 0978 0
P 0979 0
P 0980 0
P 0981 0
P 0982 0
P 0983 0
P 0984 0
P 0985 0
P 0986 0
P 0987 0
P 0988 0
P 0989 0
P 0990 0
P 0991 0
P 0992 0
P 0993 0
P 0994 0
P 0995 0
P 0996 0
P 0997 0
P 0998 0
P 0999 0
P 1000 0
P 1001 0
P 1002 0
P 1003 0
P 1004 0
P 1005 0
P 1006 0
P 1007 0
P 1008 0
P 1009 0

```
(TENNCESL, (ZX'01F7A8C4'))
(TENCUPFIL, (ZX'01F7A8CC'))
(TENNETCAP, (ZX'01F7A8D4'))
(TENTSKNA, (ZX'01F7A8DC'))
(TENNODNF, (ZX'01F7A8E4'))
(TENSFDREN, (ZX'01F7A8EC'))
(TENCDFNDR, (ZX'01F7A8F4'))
(TENJCREAD, (ZX'01F7A8FC'))
(TENRENSFD, (ZX'01F7A904'))
(TENDEVOWN, (ZX'01F7A90C'))
(TENDEVRES, (ZX'01F7A914'))
(TENDEVMDA, (ZX'01F7A91C'))
(TENDEVAL, (ZX'01F7A924'))
(TENILLDM, (ZX'01F7A92C'))
(TENLPTPAG, (ZX'01F7A934'))
(TENLPTVFU, (ZX'01F7A93C'))
(TENLPTCHR, (ZX'01F7A944'))
(TENLPTRAM, (ZX'01F7A94C'))
(TENFILSPC, (ZX'01F7A954'))
! MICCODE = 0453 (OCTAL)
(TENNSNOD, (ZX'01F7A95C'))
(TENANODI, (ZX'01F7A964'))
(TENSNODI, (ZX'01F7A96C'))
(TENNSDEV, (ZX'01F7A974'))
(TENADEVI, (ZX'01F7A97C'))
(TENSDEVI, (ZX'01F7A984'))
(TENNSDIR, (ZX'01F7A98C'))
(TENADIRI, (ZX'01F7A994'))
(TENSDIRI, (ZX'01F7A99C'))
(TENNSFIL, (ZX'01F7A9A4'))
(TENAFILI, (ZX'01F7A9AC'))
(TENSFILI, (ZX'01F7A9B4'))
(TENAFILR, (ZX'01F7A9BC'))
(TENSFILR, (ZX'01F7A9C4'))
```

DEFINE FAL STATUS CODES THAT ARE ASSOCIATED WITH THE RMS\$_SUPPORT COMPLETION CODE AND RETURNED IN THE STV FIELD OF THE FAB OR RAB.

USE THE FOLLOWING FORMULA TO MAP A DAP STATUS CODE (STSCODE FIELD WITH MACCODE = 2) INTO A CORRESPONDING FAL MESSAGE CODE (LOWER 16 BITS):

MESSAGE-CODE = (2*15 + ((MICCODE + 2048) * 8) + SEVERITY-LEVEL)

NOTE: THE INTENT IN THIS SECTION IS TO DEFINE FAL STATUS CODES ONLY FOR DAP FIELDS THAT CORRESPOND DIRECTLY TO RMS CONTROL BLOCK FIELDS USED FOR INPUT. FOR OTHER DAP FIELDS THAT MAY BE REJECTED BY THE REMOTE FAL AS BEING UNSUPPORTED, THE RMS\$_SUP COMPLETION CODE IS RETURNED IN THE STS FIELD OF THE FAB OR RAB WITH AN ASSOCIATED DAP CODE IN THE STV FIELD.

```
(OFFSET B, 2048)
! MICCODE = 0222 (OCTAL)
(ORG, (ZX'01F7C494'))
(RFM, (ZX'01F7C49C'))
(RAT, (ZX'01F7C4A4'))
(BLS, (ZX'01F7C4AC'))
(MRS, (ZX'01F7C4B4'))
(ALQ1, (ZX'01F7C4BC'))
```


P 1010	0	.(BKS,	(XX'01F7C4C4'))
P 1011	0	.(FSZ,	(XX'01F7C4CC'))
P 1012	0	.(MRN,	(XX'01F7C4D4'))
P 1013	0	! MICCODE = 0234 (OCTAL)	
P 1014	0	.(DEQ1,	(XX'01F7C4E4'))
P 1015	0	.(FOP1,	(XX'01F7C4EC'))
P 1016	0	! MICCODE = 0241 (OCTAL)	
P 1017	0	.(LRL,	(XX'01F7C50C'))
P 1018	0	! MICCODE = 0320 (OCTAL)	
P 1019	0	.(ACCFUNC,	(XX'01F7C684'))
P 1020	0	! MICCODE = 0323 (OCTAL)	
P 1021	0	.(FAC,	(XX'01F7C69C'))
P 1022	0	.(SHR,	(XX'01F7C6A4'))
P 1023	0	! MICCODE = 0420 (OCTAL)	
P 1024	0	.(CTLFUNC,	(XX'01F7C884'))
P 1025	0	! MICCODE = 0422 (OCTAL)	
P 1026	0	.(RAC,	(XX'01F7C894'))
P 1027	0	.(KEY,	(XX'01F7C89C'))
P 1028	0	.(KRF,	(XX'01F7C8A4'))
P 1029	0	.(ROP,	(XX'01F7C8AC'))
P 1030	0	! MICCODE = 0520 (OCTAL)	
P 1031	0	.(CONFUNC,	(XX'01F7CA84'))
P 1032	0	! MICCODE = 0720 (OCTAL)	
P 1033	0	.(CMPFUNC,	(XX'01F7CE84'))
P 1034	0	.(FOP2,	(XX'01F7CE8C'))
P 1035	0	! MICCODE = 1221 (OCTAL)	
P 1036	0	.(FLG,	(XX'01F7D48C'))
P 1037	0	.(DFL,	(XX'01F7D494'))
P 1038	0	.(IFL,	(XX'01F7D49C'))
P 1039	0	! MICCODE = 1225 (OCTAL)	
P 1040	0	.(POS,	(XX'01F7D4AC'))
P 1041	0	.(SIZ,	(XX'01F7D4B4'))
P 1042	0	.(REF,	(XX'01F7D4BC'))
P 1043	0	.(KNM,	(XX'01F7D4C4'))
P 1044	0	.(NUL,	(XX'01F7D4CC'))
P 1045	0	.(IAN,	(XX'01F7D4D4'))
P 1046	0	.(LAN,	(XX'01F7D4DC'))
P 1047	0	.(DAN,	(XX'01F7D4E4'))
P 1048	0	.(DTP,	(XX'01F7D4EC'))
P 1049	0	! MICCODE = 1321 (OCTAL)	
P 1050	0	.(VOL,	(XX'01F7D68C'))
P 1051	0	.(ALN,	(XX'01F7D694'))
P 1052	0	.(AOP,	(XX'01F7D69C'))
P 1053	0	.(LOC,	(XX'01F7D6A4'))
P 1054	0	! MICCODE = 1326 (OCTAL)	
P 1055	0	.(ALQ2,	(XX'01F7D6B4'))
P 1056	0	.(AID,	(XX'01F7D6BC'))
P 1057	0	.(BKZ,	(XX'01F7D6C4'))
P 1058	0	.(DEQ2,	(XX'01F7D6CC'))
P 1059	0	! MICCODE = 1521 (OCTAL)	
P 1060	0	.(CDT,	(XX'01F7DA8C'))
P 1061	0	.(RDT,	(XX'01F7DA94'))
P 1062	0	.(EDT,	(XX'01F7DA9C'))
P 1063	0	.(RVN,	(XX'01F7DAA4'))
P 1064	0	! MICCODE = 1621 (OCTAL)	
P 1065	0	.(OWNER,	(XX'01F7DC8C'))
P 1066	0	.(PROTSYS,	(XX'01F7DC94'))

P 1067 0
 P 1068 0
 P 1069 0
 1070 0
 1071 0
 1072 0
 1073 0
 1074 0
 1075 0
 1076 0
 1077 0
 1078 0
 1079 0
 1080 0
 1081 0
 1082 0
 1083 0
 1084 0
 1085 0
 1086 0
 1087 0
 1088 0
 1089 0
 1090 0
 1091 0
 1092 0
 1093 0
 1094 0
 1095 0
 1096 0
 1097 0
 1098 0
 1099 0
 1100 0
 1101 0
 1102 0
 1103 0
 1104 0
 1105 0
 1106 0
 1107 0
 1108 0
 1109 0
 1110 0
 1111 0
 1112 0
 1113 0
 1114 0
 1115 0
 1116 0
 1117 0
 1118 0
 1119 0
 1120 0
 1121 0
 1122 0
 1123 0

```
(PROTOWN,      (%X'01F7DC9C'))
(PROTGRP,      (%X'01F7DCA4'))
(PROTWLD,      (%X'01F7DCAC'))
;
```

```
*****
*
* Copyright (c) 1982, 1983
* by DIGITAL Equipment Corporation, Maynard, Mass.
*
* This software is furnished under a license and may be used and copied
* only in accordance with the terms of such license and with the
* inclusion of the above copyright notice. This software or any other
* copies thereof may not be provided or otherwise made available to any
* other person. No title to and ownership of the software is hereby
* transferred.
*
* The information in this software is subject to change without notice
* and should not be construed as a commitment by DIGITAL Equipment
* Corporation.
*
* DIGITAL assumes no responsibility for the use or reliability of its
* software on equipment which is not supplied by DIGITAL.
*
```

```
*****
*****
Created 15-SEP-1984 22:54:35 by VAX-11 SDL V2.0 Source: 15-SEP-1984 22:49:24 _$255$DUA28:[RMS.SRC]RMSINTS
*****
*****
```

*** MODULE \$IFBDEF ***

NOTE: The fields thru JNLBDB inclusive are common between the ifb and irb

```
literal IFBSC_BID = 11;          ! ifab id code
literal IFBSM_PUT = 1;
literal IFBSM_GET = 2;
literal IFBSM_DEL = 4;
literal IFBSM_UPD = 8;
literal IFBSM_TRN = 16;
literal IFBSM_BIO = 32;
literal IFBSM_BRO = 64;
literal IFBSM_EXE = 128;
literal IFBSC_SEQ = 0;          ! sequential
literal IFBSC_REL = 1;          ! relative
literal IFBSC_IDX = 2;          ! indexed
literal IFBSC_DIR = 3;          ! direct
literal IFBSC_MAXORG = 2;       ! release 1.5 maximum
literal IFBSK_FHAEND = 102;     ! end of file header attributes
literal IFBSC_FHAEND = 102;     ! end of file header attributes
literal IFBSC_KBUFNUM = 6;      ! constant - the number of key buffers allocated
literal IFBSM_ONLY_RU = 1;
literal IFBSM_RU = 2;
literal IFBSM_BI = 4;
literal IFBSM_AI = 8;
literal IFBSM_AT = 16;
```

```

1124 0 literal IFBSM_NEVER_RU = 32;
1125 0 literal IFBSM_RU_RECVR = 1;
1126 0 literal IFBSM_AI_RECVR = 2;
1127 0 literal IFBSM_BI_RECVR = 4;
1128 0 literal IFBSM_VACID_AT = 1;
1129 0 literal IFBSM_JNL = 2;
1130 0 literal IFBSM_RUP = 4;
1131 0 literal IFBSM_RU_RLK = 8;
1132 0 literal IFBSM_DONE_ASS_JNL = 16;
1133 0 literal IFBSK_BLN_SEQ = 172;
1134 0 literal IFBSC_BLN_SEQ = 172;
1135 0 --
1136 0
1137 0 organization-dependent fields
1138 0
1139 0 the following fields are used differently
1140 0 depending upon the file's organization
1141 0
1142 0 ++
1143 0
1144 0 relative org specific fields
1145 0
1146 0 literal IFBSS_IFBDEF = 172;
1147 0 (but have definitions that allow them to
1148 0 be referenced from the start of the ifab)
1149 0 ++
1150 0 the following bits are defined in
1151 0 common with the irab
1152 0
1153 0 macro IFBSV_BUSY = 4,0,1,0 %; | stream busy
1154 0 macro IFBSV_EOF = 4,1,1,0 %; | file positioned at eof
1155 0 macro IFBSV_PPF_IMAGE = 4,2,1,0 %; | flag for indirect processing of process-
1156 0 | permanent files (restricts allowable operations)
1157 0 macro IFBSV_ASYNC = 4,3,1,0 %; | async i/o flag (must be zero for ifab)
1158 0 macro IFBSV_ASYNCWAIT = 4,4,1,0 %; | wait on async i/o (must be zero for ifab)
1159 0 --
1160 0
1161 0 ifab specific bits
1162 0
1163 0 macro IFBSV_ACCESSED = 4,5,1,0 %; | file is accessed
1164 0 macro IFBSV_ANSI_D = 4,6,1,0 %; | ansi d variable records
1165 0 macro IFBSV_RWC = 4,7,1,0 %; | copy of fop bit from open
1166 0 macro IFBSV_DMO = 4,8,1,0 %; | copy of fop bit from open
1167 0 macro IFBSV_SPL = 4,9,1,0 %; | copy of fop bit from open
1168 0 macro IFBSV_SCF = 4,10,1,0 %; | copy of fop bit from open
1169 0 macro IFBSV_DLT = 4,11,1,0 %; | copy of fop bit from open
1170 0 macro IFBSV_DFW = 4,12,1,0 %; | deferred write (copy of fop bit from $open)
1171 0 macro IFBSV_SQO = 4,13,1,0 %; | sequential operations only
1172 0 macro IFBSV_PPF_INPUT = 4,14,1,0 %; | this is command 'input' stream
1173 0 macro IFBSV_NFS = 4,15,1,0 %; | non-file structured flag
1174 0 macro IFBSV_WRTACC = 4,16,1,0 %; | logical or of fac bits:
1175 0 | put, upd, del, trn
1176 0 macro IFBSV_MSE = 4,17,1,0 %; | multi-streams enabled
1177 0 macro IFBSV_CREATE = 4,18,1,0 %; | set if doing create (may be 'create if')
1178 0 macro IFBSV_NORECLK = 4,19,1,0 %; | record locking not required
1179 0 | (i.e., no shared access or multi-stream)
1180 0 macro IFBSV_RW_ATTR = 4,20,1,0 %; | set if file attributes must be re-written

```

```

1181 0 macro IFBSV_TMP = 4,21,1,0 %; temporary file (i.e., no directory entry)
1182 0 macro IFBSV_TEF = 4,22,1,0 %; truncate at eof due to large auto extend
1183 0 macro IFBSV_STALL_LOCK = 4,23,1,0 %; RMS is stalled for file lock
1184 0 macro IFBSV_SEQFIC = 4,24,1,0 %; this is really a sequential file being shared
1185 0 macro IFBSV_SEARCH = 4,25,1,0 %; search ifab - left during wildcard operations
1186 0 macro IFBSV_RMS_STALL = 4,26,1,0 %; RMS is stalled on this file operation
1187 0 macro IFBSV_RESTART = 4,27,1,0 %; Reopen or recreate operation in progress
1188 0 macro IFBSV_FILEFOUND = 4,28,1,0 %; A file was found on a search operation
1189 0 macro IFBSV_DAP_OPEN = 4,29,1,0 %; open/create function was performed via dap
1190 0 macro IFBSV_DAP = 4,30,1,0 %; data access protocol transmission
1191 0 macro IFBSV_NSP = 4,31,1,0 %; network services protocol transmission
1192 0 macro IFBSL_PRIM_DEV = 0,0,32,0 %; device characteristics bits
1193 0 ! (for primary device - bit encoding same as for fab)
1194 0 macro IFBSL_BKPBITS = 4,0,32,0 %; bookkeeping bits
1195 0
1196 0 macro IFBSB_BID = 8,0,8,0 %; block id
1197 0 macro IFBSB_BLN = 9,0,8,0 %; block length in longwords
1198 0 macro IFBSB_MODE = 10,0,8,0 %; caller's mode
1199 0 macro IFBSB_EFN = 11,0,8,0 %; event flag used for synchronous qio
1200 0 macro IFBSL_IOS = 12,0,32,0 %; internal i/o status block
1201 0 macro IFBSL_BWB = 12,0,32,0 %; bucket wait block for inter stream waiting
1202 0 macro IFBSW_IOS2 = 14,0,16,0 %; high word of io status block
1203 0 macro IFBSL_IOS4 = 16,0,32,0 %; 2nd longword of io status block
1204 0 macro IFBSL_ASBADDR = 20,0,32,0 %; address of asynchronous context block
1205 0 macro IFBSL_ARGLIST = 24,0,32,0 %; user call parameters addr
1206 0 macro IFBSL_IRAB_LNK = 28,0,32,0 %; pointer to irab(s)
1207 0 macro IFBSW_CHNL = 32,0,16,0 %; i/o channel number
1208 0 macro IFBSB_FAC = 34,0,8,0 %; file access
1209 0 macro IFBSV_PUT = 34,0,1,0 %; (same as in fab's fac field)
1210 0 macro IFBSV_GET = 34,1,1,0 %;
1211 0 macro IFBSV_DEL = 34,2,1,0 %;
1212 0 macro IFBSV_UPD = 34,3,1,0 %;
1213 0 macro IFBSV_TRN = 34,4,1,0 %;
1214 0 macro IFBSV_BIO = 34,5,1,0 %;
1215 0 macro IFBSV BRO = 34,6,1,0 %;
1216 0 macro IFBSV_EXE = 34,7,1,0 %;
1217 0 ! note: if both bio and bro set, implies block i/o
1218 0 ! access only allowed for this connect, resets
1219 0 ! to bro on disconnect (seq. file org. only).
1220 0
1221 0 macro IFBSB_ORGCASE = 35,0,8,0 %; copy of org for case dispatching
1222 0 macro IFBSL_LAST_FAB = 36,0,32,0 %; address of fab for last operation
1223 0 macro IFBSW_IFI = 40,0,16,0 %; Internal file Identifier, the one we gave to the user
1224 0 macro IFBSW_ECHO_ISI = 42,0,16,0 %; ISI of stream to echo records from SYS$INPUT
1225 0 macro IFBSL_ATJNBUF = 44,0,32,0 %; address of IFAB audit trail buffer
1226 0 macro IFBSL_JNLBDB = 48,0,32,0 %; address of Journaling BDB for FAB operations
1227 0 ! -----*****
1228 0 macro IFBSL_EXTJNLBUF = 52,0,32,0 %; pointer to buffer to contain extend journal record
1229 0 macro IFBSL_FWA_PTR = 56,0,32,0 %; pointer to file work area control block
1230 0 macro IFBSL_NWA_PTR = 60,0,32,0 %; pointer to network work area control block
1231 0 macro IFBSL_BDB_FLNK = 64,0,32,0 %; pointer to bdb(s)
1232 0 macro IFBSL_BDB_BLNK = 68,0,32,0 %; bdb backward link
1233 0 macro IFBSL_DEVBUFFSIZ = 72,0,32,0 %; device default (or bls if mt) buff size
1234 0 macro IFBSW_RTDEQ = 76,0,16,0 %; run-time default extend quantity
1235 0 macro IFBSB_SHR = 78,0,8,0 %; File sharing bits from users FAB
1236 0 macro IFBSB_AGFNT_MODE = 79,0,8,0 %; User's FABSV_FILE_MODE field, maximized with mode of caller
1237 0

```

```

1238 0      *****
1239 0
1240 0      the following fields must remain as is since
1241 0      they correspond to the rms attributes stored
1242 0      in the file header
1243 0
1244 0      macro IFBSB_RFMORG = 80,0,8,0 %;      ! organization and record format
1245 0      macro IFBSV_RFM = 80,0,4,0 %;
1246 0      literal IFBS$ RFM = 4;      ! record format (n.b. constant values defined in rfm field of fab)
1247 0      macro IFBSV_ORG = 80,4,4,0 %;
1248 0      literal IFBS$ ORG = 4;      ! file organization
1249 0      macro IFBSB_RAT = 81,0,8,0 %;      ! record attributes (n.b. bit offsets defined in rat field of fab)
1250 0      macro IFBSW_LRL = 82,0,16,0 %;      ! longest record's length (or fixed record length)
1251 0      macro IFBSL_HBK_DISK = 84,0,32,0 %;      ! hi vbn allocated (note: disk format!)
1252 0      macro IFBSL_EBK_DISK = 88,0,32,0 %;      ! eof vbn (note: disk format!)
1253 0      macro IFBSW_FFB = 92,0,16,0 %;      ! first free byte in eof block
1254 0      macro IFBSB_BKS = 94,0,8,0 %;      ! bucket size (! vbns)
1255 0      macro IFBSB_FSZ = 95,0,8,0 %;      ! record header size for vfc
1256 0      macro IFBSW_MRS = 96,0,16,0 %;      ! max record size allowable
1257 0      macro IFBSW_DEQ = 98,0,16,0 %;      ! default extend quantity
1258 0      macro IFBSW_GBC = 100,0,16,0 %;      ! global buffer count
1259 0      ! -----
1260 0      macro IFBSB_DRT_REHIT = 104,0,8,0 %;      ! hit count for local dirty buffers.
1261 0      macro IFBSB_GBL_REHIT = 105,0,8,0 %;      ! rehit count for gbl buffers.
1262 0      macro IFBSL_RNS_LEN = 108,0,32,0 %;      ! resultant name string length (used as a temp field by $search)
1263 0      macro IFBSL_LOCK_BDB = 108,0,32,0 %;      ! lock bdb address (used by $extend for rel. file)
1264 0      macro IFBSL_HBK = 112,0,32,0 %;      ! hi vbn allocated.
1265 0      macro IFBSL_EBK = 116,0,32,0 %;      ! eof vbn.
1266 0      macro IFBSL_SFSB_PTR = 120,0,32,0 %;      ! pointer to shared file synchronization block
1267 0      macro IFBSL_GBSB_PTR = 124,0,32,0 %;      ! pointer to global buffer synchronization block.
1268 0      macro IFBSL_PAR_LOCK_ID = 128,0,32,0 %;      ! Parent lock ID for bucket locks (get from SFSB.)
1269 0      macro IFBSW_AVLCL = 132,0,16,0 %;      ! local buffers available.
1270 0      macro IFBSW_AVGBPB = 134,0,16,0 %;      ! gbl ptr blocks available.
1271 0      macro IFBSL_GBH_PTR = 136,0,32,0 %;      ! pointer to global header.
1272 0      macro IFBSL_AS_DEV = 140,0,32,0 %;      ! assigned device characteristics
1273 0      ! AS DEV and ASDEVBSIZ *)
1274 0      macro IFBSL_ASDEVBSIZ = 148,0,32,0 %;      ! assigned device buffer size
1275 0      macro IFBSL_BLBFLNK = 152,0,32,0 %;      ! Forward link to BLB chain.
1276 0      macro IFBSL_BLBBLNK = 156,0,32,0 %;      ! Back link to BLB chain.
1277 0      macro IFBSB_JNLFLG = 160,0,8,0 %;      ! journaling attribute flags
1278 0      macro IFBSV_ONLY_RU = 160,0,1,0 %;      ! Recovery Unit journaling, no access outside RU
1279 0      macro IFBSV_RU = 160,1,1,0 %;      ! Recovery Unit journaling
1280 0      macro IFBSV_BI = 160,2,1,0 %;      ! Before Image journaling
1281 0      macro IFBSV_AI = 160,3,1,0 %;      ! After Image journaling
1282 0      macro IFBSV_AT = 160,4,1,0 %;      ! Audit Trail journaling
1283 0      macro IFBSV_NEVER_RU = 160,5,1,0 %;      ! never do RU journaling
1284 0      macro IFBSB_RECVRFLGS = 161,0,8,0 %;      ! Recovery flags
1285 0      macro IFBSV_RU_RECVR = 161,0,1,0 %;      ! Recovery Unit Rollback in progress
1286 0      macro IFBSV_AI_RECVR = 161,1,1,0 %;      ! AI Roll Forward Recovery in progress
1287 0      macro IFBSV_BI_RECVR = 161,2,1,0 %;      ! BI Roll Backward Recovery in progress
1288 0      macro IFBSB_JNLFLG2 = 162,0,8,0 %;      ! Secondary journaling flags (generally operation specific)
1289 0      macro IFBSV_VALID_AT = 162,0,1,0 %;      ! AT entry in IFB buffer is valid and should be written
1290 0      macro IFBSV_JNL = 162,1,1,0 %;      ! Journaling Initialized for this file
1291 0      macro IFBSV_RUP = 162,2,1,0 %;      ! Recovery Unit in progress
1292 0      macro IFBSV_RU_RLK = 162,3,1,0 %;      ! Fake record locking during recovery unit
1293 0      macro IFBSV_DONE_ASS_JNL = 162,4,1,0 %;      ! Journal channels already assigned
1294 0      macro IFBSL_RJB = 164,0,32,0 %;      ! RMS Journaling Block address

```

```

1295 0 macro IFB$W_BUFFER_OFFSET = 168,0,16,0 %; ! ANSI buffer offset
1296 0 literal IFB$K_BLN_REL = 180;
1297 0 literal IFB$C_BLN_REL = 180;
1298 0 --
1299 0 ++
1300 0
1301 0 indexed org specific fields
1302 0
1303 0 literal IFB$S_IFBDEF1 = 180;
1304 0 macro IFB$L_MRN = 172,0,32,0 %; ! (rel) max record number
1305 0 macro IFB$L_DVBN = 176,0,32,0 %; ! (rel) first data bucket vbn
1306 0 literal IFB$K_BLN_IDX = 184;
1307 0 literal IFB$C_BLN_IDX = 184;
1308 0 literal IFB$K_BLN = 184; ! ifab length
1309 0 literal IFB$C_BLN = 184; ! ifab length
1310 0 --
1311 0 literal IFB$S_IFBDEF2 = 184;
1312 0 macro IFB$L_IDX_PTR = 172,0,32,0 %; ! (idx) pointer to primary key index descriptor
1313 0 macro IFB$B_AVBN = 176,0,8,0 %; ! (idx) vbn of 1st area descriptor
1314 0 macro IFB$B_AMAX = 177,0,8,0 %; ! (idx) total number of area descriptors
1315 0 macro IFB$B_NUM_KEYS = 178,0,8,0 %; ! (idx) ! of keys in file
1316 0 macro IFB$B_UBUFSZ = 179,0,8,0 %; ! (idx) update buffer size for keys
1317 0 macro IFB$W_KBUFSZ = 180,0,16,0 %; ! (idx) key buffer size
1318 0 macro IFB$B_EXTRABUF = 182,0,8,0 %; ! (idx) number of extra buffers for 'cache'ing
1319 0 macro IFB$B_PLG_VER = 183,0,8,0 %; ! (idx) prologue version number
1320 0
1321 0 *** MODULE $IRBDEF ***
1322 0
1323 0 IRB field definitions
1324 0
1325 0 Internal rab (irb)
1326 0
1327 0 There is 1 irab per connected record access stream
1328 0
1329 0
1330 0 NOTE: The fields thru JNLBDB inclusive are common between the irb and ifb
1331 0
1332 0 literal IRB$C_BID = 10; ! irab code
1333 0 literal IRB$M_POSINSERT = 1;
1334 0 literal IRB$M_SRCHGT = 2;
1335 0 literal IRB$M_POSDELETE = 4;
1336 0 literal IRB$M_NEW_IDX = 8;
1337 0 literal IRB$M_SRCRGE = 16;
1338 0 literal IRB$M_NORLS_RNF = 32;
1339 0 literal IRB$M_FIRST_TIM = 64;
1340 0 literal IRB$M_PRM = 128;
1341 0 literal IRB$M_DUP_KEY = 256;
1342 0 literal IRB$M_DEL_SEEN = 512;
1343 0 literal IRB$M_LAST_GT = 1024;
1344 0 literal IRB$M_BKT_NO_LO = 1;
1345 0 literal IRB$M_NEW_BKTS = 6;
1346 0 literal IRB$M_REC_W_LO = 8;
1347 0 literal IRB$M_CONT_BKT = 16;
1348 0 literal IRB$M_CONT_R = 32;
1349 0 literal IRB$M_EMPTY_BKT = 64;
1350 0 literal IRB$M_DUPS_SEEN = 128;
1351 0 literal IRB$M_BKT_NO = 3;

```

```

1352 0 literal IRBSM_BIG_SPLIT = 4;
1353 0 literal IRBSM_SPLIT_IDX = 1;
1354 0 literal IRBSM_EMPTY_SEEN = 2;
1355 0 literal IRBSM_VALID_AT = 1;
1356 0 literal IRBSK_BLN_REL = 100;
1357 0 literal IRBSC_BLN_REL = 100;
1358 0 ++
1359 0
1360 0 sequential org specific fields
1361 0
1362 0 literal IRBSK_BLN_SEQ = 108;
1363 0 literal IRBSC_BLN_SEQ = 108;
1364 0 literal IRBSS_IRBDEF = 108;
1365 0 to apply from start of irab
1366 0 ++
1367 0
1368 0 the following bits are defined in common
1369 0 with the ifab
1370 0
1371 0 macro IRBSV_BUSY = 4,0,1,0 %; | file busy
1372 0 macro IRBSV_EOF = 4,1,1,0 %; | stream positioned at eof
1373 0 macro IRBSV_PPF_IMAGE = 4,2,1,0 %; | flag for indirect processing of process-
1374 0 permanent file
1375 0 macro IRBSV_ASYNC = 4,3,1,0 %; | asynchronous i/o request
1376 0 macro IRBSV_ASYNCWAIT = 4,4,1,0 %; | $wait issued for asynchronous i/o request
1377 0 --
1378 0
1379 0 irab specific bits
1380 0
1381 0 macro IRBSV_FIND_LAST = 4,5,1,0 %; | last operation was a find
1382 0 macro IRBSV_PUTS_LAST = 4,6,1,0 %; | last operation was a put sequential
1383 0 macro IRBSV_BIO_LAST = 4,7,1,0 %; | this/last operation is/was a block i/o operation
1384 0 note: this bit is set only if mixed block and record
1385 0 operations (bro access). after call to rmsrset
1386 0 refers to the current operation and bro_sw gives
1387 0 type of last operation.
1388 0 macro IRBSV_BRO_SW = 4,8,1,0 %; | switched from record operation to block i/o operation
1389 0 macro IRBSV_FIND = 4,9,1,0 %; | operation is a find
1390 0 macro IRBSV_RAHEAD = 4,10,1,0 %; | read ahead or write behind processing
1391 0 macro IRBSV_SKIP_NEXT = 4,11,1,0 %; | skip to next record flag for index fo
1392 0 macro IRBSV_DUP = 4,12,1,0 %; | duplicate records seen
1393 0 macro IRBSV_UNLOCK_RP = 4,13,1,0 %; | release lock on current (rp) record
1394 0 macro IRBSV_PPF_EOF = 4,14,1,0 %; | give one-shot rms_eof error on sys$input
1395 0 macro IRBSV_PPF_SKIP = 4,15,1,0 %; | skip sys$input record ($deck), redoing $get
1396 0 ! or $find on next record
1397 0 macro IRBSV_PPF_FNDV = 4,16,1,0 %; | save value for find bit when ppf_skip set
1398 0 macro IRBSV_IDX_ERR = 4,17,1,0 %; | index update error occurred
1399 0 macro IRBSV_RRV_ERR = 4,18,1,0 %; | rrv update error occurred
1400 0 macro IRBSV_UPDATE = 4,19,1,0 %; | operation is an update (indexed)
1401 0 macro IRBSV_UPDATE_IF = 4,20,1,0 %; | operation was a $PUT -> $UPDATE
1402 0 macro IRBSV_ABOVELOCKD = 4,21,1,0 %; | level above was locked by search_tree
1403 0 macro IRBSV_GBLBUFF = 4,22,1,0 %; | global buffers are in use.
1404 0 macro IRBSV_CON_EOF = 4,23,1,0 %; | file positioned at EOF by $CONNECT (isam)
1405 0 macro IRBSV_NO_WAIT = 4,24,1,0 %; | do not wait for enqueues on query locks
1406 0 macro IRBSV_PPF_ECHO = 4,25,1,0 %; | echo SYS$INPUT records to SYS$OUTPUT
1407 0 macro IRBSV_RMS_STALL = 4,26,1,0 %; | RMS is stalled on this record operation
1408 0 macro IRBSV_RESTART = 4,27,1,0 %; | Reconnect operation in progress

```

```

1409 0 macro IRBSV_DAP_CONN = 4,28,1,0 %;      | connect function was performed via dap
1410 0 macro IRBSV_RU_DELETE = 4,29,1,0 %;      | recovery unit deletion in progress
1411 0 macro IRBSV_RU_UNDEL = 4,30,1,0 %;      | recovery unit un-deletion in progress
1412 0 macro IRBSV_RU_UPDATE = 4,31,1,0 %;      | place new record is special RU UPDATE format
1413 0
1414 0 the following are alternate definitions for alternate
1415 0 (non-conflicting) use of the above bits
1416 0
1417 0 macro IRBSV_WRITE = 4,9,1,0 %;      | operation is a write
1418 0 macro IRBSL_IFAB_LNK = 0,0,32,0 %;      | pointer to ifab
1419 0 macro IRBSL_BKPBITS = 4,0,32,0 %;      | bookkeeping status bits
1420 0
1421 0 macro IRBSB_UID = 8,0,8,0 %;      | block id
1422 0 macro IRBSB_BLN = 9,0,8,0 %;      | block length in longwords
1423 0 macro IRBSB_MODE = 10,0,8,0 %;      | caller's mode
1424 0 macro IRBSB_EFN = 11,0,8,0 %;      | event flag for synchronous io
1425 0 macro IRBSL_IOS = 12,0,32,0 %;      | internal i/o status block
1426 0 macro IRBSL_BWB = 12,0,32,0 %;      | bucket wait block for inter stream locking
1427 0 macro IRBSW_IOS2 = 14,0,16,0 %;      | high word of io status block
1428 0 macro IRBSL_IOS4 = 16,0,32,0 %;      | io status block (2nd longword)
1429 0 macro IRBSL_ASBADDR = 20,0,32,0 %;      | address of permanent asynchronous context block
1430 0 macro IRBSL_ARGLIST = 24,0,32,0 %;      | user arg list address
1431 0 ! if async, points to copy at head
1432 0 ! of async context block
1433 0 macro IRBSL_IRAB_LNK = 28,0,32,0 %;      | pointer to next irab
1434 0 macro IRBSL_CURBDB = 32,0,32,0 %;      | current bdb address
1435 0 macro IRBSL_LAST_RAB = 36,0,32,0 %;      | address of rab for last operation
1436 0 macro IRBSW_ISI = 40,0,16,0 %;      | Internal stream Identifier, the one we gave to the user
1437 0 macro IRBSL_ATJNLBUF = 44,0,32,0 %;      | address of IRAB auuit trail journaling buffer
1438 0 macro IRBSL_JNLBDB = 48,0,32,0 %;      | address of journaling BDB for RAB operations
1439 0 ! -----
1440 0 macro IRBSL_IDENT = 52,0,32,0 %;      | process unique identifier for the IRB
1441 0 macro IRBSL_RLB_LNK = 56,0,32,0 %;      | pointer to RLBS
1442 0 macro IRBSL_NXTBDB = 60,0,32,0 %;      | next bdb address
1443 0 macro IRBSL_NRP = 64,0,32,0 %;      | next record pointer (relative record number)
1444 0 macro IRBSL_NRP_VBN = 64,0,32,0 %;      | next record pointer (relative)
1445 0 macro IRBSB_CACHEFLGS = 64,0,8,0 %;      | cache flags for calls to getbkt, cache, etc. (indexed)
1446 0 macro IRBSB_STOPLEVEL = 65,0,8,0 %;      | level to stop at on tree search (indexed)
1447 0 macro IRBSW_SRCHFLAGS = 66,0,16,0 %;      | search flags (indexed)
1448 0 macro IRBSV_POSINSERT = 66,0,1,0 %;      | position for insert
1449 0 macro IRBSV_SRCHGT = 66,1,1,0 %;      | approximate search gt
1450 0 macro IRBSV_POSDELETE = 66,2,1,0 %;      | position for delete
1451 0 macro IRBSV_NEW_IDX = 66,3,1,0 %;      | need to read in new idx dsc from file
1452 0 macro IRBSV_SRCHGE = 66,4,1,0 %;      | approximate search ge
1453 0 macro IRBSV_NORLS_RNF = 66,5,1,0 %;      | don't release bkt on rnf error, if set
1454 0 macro IRBSV_FIRST_TIM = 66,6,1,0 %;      | flag to indicate 1st time for seq. processing
1455 0 macro IRBSV_PRM = 66,7,1,0 %;      | flag to indicate that the permanence bit in the bdb
1456 0 macro IRBSV_DUP_KEY = 66,8,1,0 %;      | a duplicate key seen on scan of any data bucket
1457 0 macro IRBSV_DEL_SEEN = 66,9,1,0 %;      | a deleted record has been encountered between current
1458 0 macro IRBSV_LAST_GT = 66,10,1,0 %;      | result of last search of compressed key bucket was GT
1459 0 ! and a next record during a $GET/$FIND
1460 0 ! should be set
1461 0 macro IRBSL_NRP_OFF = 68,0,32,0 %;      | next record pointer offset (relative)
1462 0 macro IRBSL_CURVBN = 68,0,32,0 %;      | ybn of current record (relative)
1463 0 macro IRBSW_NRP_OFF = 68,0,16,0 %;      |
1464 0 macro IRBSB_SPL_BITS = 68,0,8,0 %;      | bits for splitting (indexed)
1465 0 macro IRBSV_BKT_NO_LO = 68,0,1,0 %;      | low bit of bucket number processing

```



```

1466 0 macro IRBSV_NEW_BKTS = 68,1,2,0 %;
1467 0 literal IRBS NEW_BKTS = 2;
1468 0 macro IRBSV_REC_W_LO = 68,3,1,0 %;
1469 0 macro IRBSV_CONT_BKT = 68,4,1,0 %;
1470 0 macro IRBSV_CONT_R = 68,5,1,0 %;
1471 0 macro IRBSV_EMPTY_BKT = 68,6,1,0 %;
1472 0 macro IRBSV_DUPS_SEEN = 68,7,1,0 %;
1473 0 macro IRBSV_BKT_NO = 68,0,2,0 %;
1474 0 literal IRBS BKT_NO = 2;
1475 0 macro IRBSV_BIG_SPLIT = 68,2,1,0 %;
1476 0 macro IRBSV_SPL_IDX = 68,0,1,0 %;
1477 0 macro IRBSV_EMPTY_SEEN = 68,1,1,0 %;
1478 0 macro IRBSL_RP = 72,0,32,0 %;
1479 0 macro IRBSL_RP_VBN = 72,0,32,0 %;
1480 0 macro IRBSW_POS_INS = 72,0,16,0 %;
1481 0 macro IRBSW_SPLIT = 74,0,16,0 %;
1482 0 macro IRBSL_RP_OFF = 76,0,32,0 %;
1483 0 macro IRBSL_LST_REC = 76,0,32,0 %;
1484 0 macro IRBSL_PTR_VBN = 76,0,32,0 %;
1485 0 macro IRBSW_RP_OFF = 76,0,16,0 %;
1486 0 macro IRBSW_SPLIT_1 = 76,0,16,0 %;
1487 0 macro IRBSW_SPLIT_2 = 78,0,16,0 %;
1488 0 macro IRBSL_OWNER_ID = 80,0,32,0 %;
1489 0 macro IRBSW_OW_N_ID = 80,0,16,0 %;
1490 0 macro IRBSW_OW_N_ISI = 82,0,16,0 %;
1491 0 macro IRBSB_PPF_ISI = 82,0,8,0 %;
1492 0 macro IRBSB_BCNT = 84,0,8,0 %;
1493 0 macro IRBSB_MBC = 85,0,8,0 %;
1494 0 macro IRBSW_RSZ = 86,0,16,0 %;
1495 0 macro IRBSL_RBF = 88,0,32,0 %;
1496 0 macro IRBSB_MBF = 92,0,8,0 %;
1497 0 macro IRBSB_JNLFLG3 = 93,0,8,0 %;
1498 0 macro IRBSV_VALID_AT = 93,0,1,0 %;
1499 0 ++
1500 0
1501 0 start of organization dependent fields
1502 0
1503 0 ++
1504 0
1505 0 used by sequential and relative files
1506 0
1507 0 macro IRBSW_CSIZ = 98,0,16,0 %;
1508 0 ++
1509 0
1510 0 relative org specific fields
1511 0
1512 0 macro IRBSL_TEMPO = 100,0,32,0 %;
1513 0 macro IRBSW_ROVHDSZ = 100,0,16,0 %;
1514 0 macro IRBSB_PRE_CCTL = 100,0,8,0 %;
1515 0 macro IRBSB_POST_CCTL = 101,0,8,0 %;
1516 0 macro IRBSW_RTOTCSZ = 102,0,16,0 %;
1517 0 macro IRBSL_TEMP1 = 104,0,32,0 %;
1518 0 macro IRBSB_NVBNs = 104,0,8,0 %;
1519 0
1520 0 indexed org specific fields
1521 0
1522 0 literal IRBSK_BLN_IDX = 196;

```

```

number of new buckets (0-3)
if splitting at pos_insert than rec goes w/ lo
middle bucket is a continuation bkt
right bucket is a continuation bkt
bucket contains no data records
dups seen on scan of bucket, any key

split up new index record and swing pointer
empty bucket passed over on posinsert
record pointer (relative record !)
record pointer (relative)
offset for position for insert for put (indexed)
first split point (indexed)
record pointer offset
last record address (indexed)
pointer vbn used by find_by_rrv (indexed)
record pointer offset
second split point -- 3-bkt split (indexed)
third split point -- 4-bkt split (indexed)
owner id used for record locks
index part of process id (pid)
isi value for this irab
isi value for this process-permanent irab
i/o buffer count
multi-block count
record size from user
user record buffer address
Multi-buffer count from user's RAB
IRB journaling flags
IRB MJB contains valid AT entry to write

```

```

1523 0 literal IRB$C_BLN_IDX = 196;
1524 0 literal IRB$S_IRBDEF1 = 196;
1525 0 macro IRB$L_KEYBUF = 96,0,32,0 %;
1526 0 macro IRB$L_UPDBUF = 100,0,32,0 %;
1527 0 macro IRB$L_RECBUF = 104,0,32,0 %;
1528 0 macro IRB$L_OLDBUF = 108,0,32,0 %;
1529 0 macro IRB$L_RFA_VBN = 112,0,32,0 %;
1530 0 macro IRB$L_UPD_BDB = 112,0,32,0 %;
1531 0 macro IRB$L_LAST_VBN = 112,0,32,0 %;
1532 0 macro IRB$W_RFA_ID = 116,0,16,0 %;
1533 0 macro IRB$W_LAST_ID = 116,0,16,0 %;
1534 0 macro IRB$W_SAVE_POS = 118,0,16,0 %;
1535 0 macro IRB$L_NEXT_VBN = 120,0,32,0 %;
1536 0 macro IRB$L_PUTUP_VBN = 120,0,32,0 %;
1537 0 macro IRB$L_FIRST_VBN = 124,0,32,0 %;
1538 0 macro IRB$W_NEXT_ID = 128,0,16,0 %;
1539 0 macro IRB$W_PUTUP_ID = 128,0,16,0 %;
1540 0 macro IRB$W_FIRST_ID = 130,0,16,0 %;
1541 0 macro IRB$L_LOCK_BDB = 132,0,32,0 %;
1542 0 macro IRB$L_VBN_LEFT = 136,0,32,0 %;
1543 0 macro IRB$L_MIDX_TMP1 = 136,0,32,0 %;
1544 0 macro IRB$L_VBN_RIGHT = 140,0,32,0 %;
1545 0 macro IRB$L_MIDX_TMP2 = 140,0,32,0 %;
1546 0 macro IRB$L_VBN_MID = 144,0,32,0 %;
1547 0 macro IRB$L_MIDX_TMP3 = 144,0,32,0 %;
1548 0 macro IRB$L_NEXT_DOWN = 144,0,32,0 %;
1549 0 macro IRB$L_REC_COUNT = 148,0,32,0 %;
1550 0 macro IRB$L_LST_NCMP = 152,0,32,0 %;
1551 0 macro IRB$L_SPL_COUNT = 156,0,32,0 %;
1552 0 ! when splitting indexes and SDRs
1553 0 macro IRB$W_NID_RIGHT = 160,0,16,0 %;
1554 0 macro IRB$W_NID_MID = 162,0,16,0 %;
1555 0 macro IRB$W_RFA_NID = 164,0,16,0 %;
1556 0 macro IRB$B_KEYSZ = 166,0,8,0 %;
1557 0 macro IRB$L_CUR_VBN = 168,0,32,0 %;
1558 0 macro IRB$L_POS_VBN = 172,0,32,0 %;
1559 0 macro IRB$L_UDR_VBN = 176,0,32,0 %;
1560 0 macro IRB$L_SIDR_VBN = 180,0,32,0 %;
1561 0 macro IRB$W_CUR_ID = 184,0,16,0 %;
1562 0 macro IRB$W_POS_ID = 186,0,16,0 %;
1563 0 macro IRB$W_UDR_ID = 188,0,16,0 %;
1564 0 macro IRB$W_SIDR_ID = 190,0,16,0 %;
1565 0 macro IRB$W_CUR_COUNT = 192,0,16,0 %;
1566 0 macro IRB$B_RP_KREF = 194,0,8,0 %;
1567 0 macro IRB$B_CUR_KREF = 195,0,8,0 %;
1568 0
1569 0 *** MODULE $ASBDEF ***
1570 0
1571 0 ASB field definitions
1572 0
1573 0 Asynchronous context block (asb)
1574 0
1575 0 There is one asb per irab pointed to by irb$l_asbaddr allocated at
1576 0 connect and one per ifab which is dynamically allocated at stall
1577 0
1578 0 The asb$l_arglst is pointed to by the arglst field of the
1579 0 irab if the irb$w_async bookkeeping bit is set

```

address of internal key buffer & update buffer
 address of internal update buffer
 address of internal record buffer
 address of internal old record buffer (updates only)
 save record vbn for nrp data
 save current bdb during insert operation
 last vbn at data level for update
 save record id for search data
 id for udr during update (plg 3)
 save duplicate position for nrp data
 save next user data record VBN for nrp data
 RFA VBN of \$PUT/\$UPDATE record
 save SDR first element VBN for search NRP data
 save next user data record ID for nrp data
 ID of \$PUT/\$UPDATE record
 save SDR first element ID for search NRP data
 lock bdb addr of level below on splits
 left vbn of split
 temporary one for make index
 right vbn of split
 temporary two for make index
 middle vbn of split
 temporary three for make index
 used by search tree
 number of current record in this bucket (plg 3)
 address of last key with zero front compression (plg 3)
 number of the first record to be moved into new bucket

Next record ID of the right bucket
 Next record ID of the middle bucket
 Next record ID of the RFA bucket
 size of key in keybuffer !2
 VBN of current record (primary/SIDR)
 VBN of primary data record for NRP positioning
 VBN of current primary data record
 SDR array first element VBN of current record (SIDR)
 ID of current record (primary)
 ID of primary data record for NRP positioning
 ID of current primary data record
 SDR array first element ID of current record (SIDR)
 SDR array count of current record (SIDR)
 Key of reference by which next record is retrieved
 Key of reference of current record (primary/SIDR)

```

1580 0
1581 0      All of the asb$C_bln_xxx must be longword aligned
1582 0
1583 0      literal ASB$C_BID = 13;      ! asb id = 13
1584 0      literal ASB$K_BLN_FIX = 48; ! block length of fixed asb
1585 0      literal ASB$C_BLN_FIX = 48; ! block length of fixed asb
1586 0      ! regs 4 and 5 are saved on stack
1587 0      literal ASB$K_BLN_SEQ = 188; ! block length for seq org irab operations
1588 0      literal ASB$C_BLN_SEQ = 188; ! block length for seq org irab operations
1589 0      literal ASB$K_BLN_REL = 192; ! block length for rel org irab operations
1590 0      literal ASB$C_BLN_REL = 192; ! block length for rel org irab operations
1591 0      literal ASB$K_BLN_FAB = 352; ! block length for fab-related operations
1592 0      literal ASB$C_BLN_FAB = 352; ! block length for fab-related operations
1593 0      literal ASB$K_BLN_IDX = 512;
1594 0      literal ASB$C_BLN_IDX = 512;
1595 0      literal ASB$S_ASBDEF = 512;
1596 0      macro ASB$W_STKLEN = 0,0,16,0 %; ! save stack length (must be first word in block)
1597 0      ! STKLEN = BLN org - BLN FIX
1598 0      macro ASB$W_STKSTZ = 2,0,16,0 %; ! size of saved stack in bytes
1599 0      macro ASB$B_BID = 8,0,8,0 %; ! block id
1600 0      macro ASB$B_BLN = 9,0,8,0 %; ! block length in longwords
1601 0      macro ASB$L_ARGLST = 12,0,0,0 %;
1602 0      literal ASB$S_ARGLST = 16; ! saved argument list on async irab operations
1603 0      macro ASB$B_ARGCNT = 12,0,8,0 %; ! argument count
1604 0      ! value will be 0, 1, 2, or 3
1605 0      macro ASB$L_FABRAB = 16,0,32,0 %; ! fab or rab address
1606 0      macro ASB$L_ERR = 20,0,32,0 %; ! err routine addr
1607 0      macro ASB$L_SUC = 24,0,32,0 %; ! suc routine addr
1608 0      macro ASB$L_REGS = 28,0,0,0 %;
1609 0      literal ASB$S_REGS = 20; ! save register area for regs 6, 7, 8, 10 and 11
1610 0      macro ASB$L_STK = 48,0,0,0 %;
1611 0      literal ASB$S_STK = 140; ! saved stack area
1612 0
1613 0      !*** MODULE $BDBDEF ***
1614 0
1615 0      BDB field definitions
1616 0
1617 0      buffer descriptor block (bdb)
1618 0
1619 0      there is one bdb per i/o buffer
1620 0      ( the i/o buffers exist in separate pages, page aligned)
1621 0
1622 0      literal BDB$C_BID = 12;      ! bdb id code
1623 0      literal BDB$M_VAL = 1;
1624 0      literal BDB$M_DRT = 2;
1625 0      literal BDB$M_IOP = 4;
1626 0      literal BDB$M_PRM = 8;
1627 0      literal BDB$M_NOLOCATE = 16;
1628 0      literal BDB$M_WFO = 32;
1629 0      literal BDB$M_AST_DCL = 64;
1630 0      literal BDB$S_BDBDEF = 80;
1631 0      macro BDB$L_FLINK = 0,0,32,0 %; ! forward link
1632 0      macro BDB$L_BLINK = 4,0,32,0 %; ! backward link
1633 0      macro BDB$B_BID = 8,0,8,0 %; ! block id
1634 0      macro BDB$B_BLN = 9,0,8,0 %; ! block length in longwords
1635 0      macro BDB$B_FLGS = 10,0,8,0 %; ! bdb flags
1636 0      macro BDB$V_VAL = 10,0,1,0 %; ! buffer contents valid

```

```

1637 0 macro BDB$V_DRT = 10,1,1,0 %;
1638 0 macro BDB$V_IOP = 10,2,1,0 %;
1639 0 macro BDB$V_PRM = 10,3,1,0 %;
1640 0 macro BDB$V_NOLOCATE = 10,4,1,0 %;
1641 0 macro BDB$V_WFO = 10,5,1,0 %;
1642 0 macro BDB$V_AST_DCL = 10,6,1,0 %;
1643 0 waiting stream
1644 0 the releasing of this bdb
1645 0 (set/cleared by rm$cache)
1646 0 macro BDB$B_CACHE_VAL = 11,0,8,0 %;
1647 0 macro BDB$B_VERTYP = 11,0,8,0 %;
1648 0 macro BDB$W_USERS = 12,0,16,0 %;
1649 0 macro BDB$W_BUFF_ID = 14,0,16,0 %;
1650 0 macro BDB$L_BLB_PTR = 16,0,32,0 %;
1651 0 macro BDB$W_NUMB = 20,0,16,0 %;
1652 0 macro BDB$W_DIRSEQ = 20,0,16,0 %;
1653 0 macro BDB$W_SIZE = 22,0,16,0 %;
1654 0 macro BDB$L_ADDR = 24,0,32,0 %;
1655 0 macro BDB$L_VBN = 28,0,32,0 %;
1656 0 macro BDB$L_VBNSEQNO = 32,0,32,0 %;
1657 0 macro BDB$L_LAST = 32,0,32,0 %;
1658 0 macro BDB$L_WAIT = 36,0,32,0 %;
1659 0 (for inter-stream intra-
1660 0 process locking only)
1661 0 macro BDB$L_VERCOUNT = 36,0,32,0 %;
1662 0 macro BDB$L_ALLOC_ADDR = 40,0,32,0 %;
1663 0 macro BDB$W_ALLOC_SIZE = 44,0,16,0 %;
1664 0 macro BDB$L_BI_BDB = 48,0,32,0 %;
1665 0 macro BDB$L_AI_BDB = 52,0,32,0 %;
1666 0 macro BDB$T_JN[SEQ] = 56,0,0,0 %;
1667 0 literal BDB$S_JNLSEQ = 16;
1668 0 macro BDB$L_WRI = 72,0,32,0 %;
1669 0 macro BDB$B_REL_VBN = 72,0,8,0 %;
1670 0 macro BDB$B_VAL_VBNS = 73,0,8,0 %;
1671 0 macro BDB$B_PRE_CTL = 74,0,8,0 %;
1672 0 macro BDB$B_POST_CTL = 75,0,8,0 %;
1673 0 macro BDB$L_CURBUF_PTR = 76,0,32,0 %;
1674 0 literal BDB$K_BLN = 20;
1675 0 literal BDB$C_BLN = 80;
1676 0 literal BDB$S_BDBDEF1 = 80;
1677 0 macro BDB$L_IOSB = 72,0,0,0 %;
1678 0 literal BDB$S_IOSB = 8;
1679 0 macro BDB$L_VERSION = 72,0,32,0 %;
1680 0 macro BDB$L_RECORD = 76,0,32,0 %;

1681 0
1682 0 *** MODULE $GBPBDEF ***
1683 0
1684 0 GBP field definitions
1685 0
1686 0 Global Buffer Pointer Block (GBPB)
1687 0
1688 0 The GBPB is the process local structure used in conjunction with
1689 0 shared global i/o buffers. In order to minimize the impact of
1690 0 global buffers on existing code, the GBPB is identical to a BDB
1691 0 in those fields which are referenced outside of the RM$CACHE and
1692 0 RM$RELEASE routines.
1693 0

```

```

: buffer content dirty
: buffer has i/o in progress
: buffer has permanence factor
: buffer shared - no locate mode
: other streams awaiting
: ast has been declared for

:
:
: relative value of buffer in cache
: version type (1 = wi 1)
: number of streams referencing this buffer
: buffer identification number
: pointer to BLB chain for this BDB
: ! of bytes of buffer in use
: UCBSW_DIRSEQ at directory read time
: ! bytes in buffer
: address of buffer
: 1st vbn in buffer
: vbn seq number of validity check vs. bcb copy
: address of last directory record
: wait thread (irab addr)

:
: negative count of version entries scanned
: buffer allocation addr
: buffer allocation size
: address of isam/block i/o bi journaling BDB
: address of isam/block i/o ai journaling BDB

:
: Journaling Sequence Number Block
: work area
: current vbn rel to start of buffer
: ! of valid vbns in buffer
: unit record carriage control byte ('pre')
: unit record carriage control byte ('post')
: current buffer addr
: length of bdb block
: length of bdb block

:
: i/o status block for buffer
: addr of current/next directory version entry
: address of current/next directory record

```

```
!*** MODULE SRLBDEF ***
```

record lock block (rlb)

```

rlb:  +-----+
      |!                                     !|
      |link                               link|
      +-----+-----+-----+-----+
      |!                                     !|
      |!               rfa4 id             !|
      +-----+-----+-----+-----+
      |flags reserved| bln   | bid         |
      +-----+-----+-----+-----+
      |rfa0                                         |
      +-----+-----+-----+-----+
      |owner                                         |
      +-----+-----+-----+-----+
lksb: | Still to be def- | VMS status code
      | ined status bits |
      +-----+-----+-----+-----+
      | Lock Id. (Returned for new locks,
      | input for conversions)
      +-----+-----+-----+-----+

```

```
literal RLBSM_TIMER_INPROG = 1;
literal RLBSM_BID = 14;
literal RLBSM_WAIT = 1;
literal RLBSM_CR = 2;
literal RLBSM_PW = 4;
literal RLBSM_PR = 8;
```

```

1751 0 literal RLBSM_CONV = 16;
1752 0 literal RLBSM_LV2 = 32;
1753 0 literal RLBSM_FAKE = 64;
1754 0 literal RLBSM_TMO = 128;
1755 0 literal RLBSK_BLN = 28;
1756 0 literal RLBSK_BLN = 28;
1757 0 literal RLBS$RLBDEF = 28;
1758 0 macro RLBSL_LNK = 0,0,32,0 %;
1759 0 macro RLBSL_MISC = 4,0,32,0 %;
1760 0 macro RLBSW_FLAGS2 = 4,0,16,0 %;
1761 0 macro RLBSV_TIMER_INPROG = 4,0,1,0 %;
1762 0 macro RLBSW_RFA4 = 6,0,16,0 %;
1763 0 ! offset for seq f.o. (bits 0:14)
1764 0 ! always 0 for rel f.o. (bits 0:14)
1765 0 macro RLBSW_ID = 6,0,16,0 %;
1766 0 macro RLBSB_BID = 8,0,8,0 %;
1767 0 macro RLBSB_BLN = 9,0,8,0 %;
1768 0 macro RLBSB_TMO = 10,0,8,0 %;
1769 0 macro RLBSB_FLAGS = 11,0,8,0 %;
1770 0 macro RLBSV_WAIT = 11,0,1,0 %;
1771 0 macro RLBSV_CR = 11,1,1,0 %;
1772 0 macro RLBSV_PW = 11,2,1,0 %;
1773 0 macro RLBSV_PR = 11,3,1,0 %;
1774 0 macro RLBSV_CONV = 11,4,1,0 %;
1775 0 macro RLBSV_LV2 = 11,5,1,0 %;
1776 0 macro RLBSV_FAKE = 11,6,1,0 %;
1777 0 macro RLBSV_TMO = 11,7,1,0 %;
1778 0 ! indicate "lock for write, allow readers"
1779 0 ! used to query lock database for records
1780 0 macro RLBSL_RFA0 = 12,0,32,0 %;
1781 0 ! seq f.o. vbn
1782 0 ! rel f.o. relative record number
1783 0 ! idx f.o. start vbn
1784 0 macro RLBSL_OWNER = 16,0,32,0 %;
1785 0 macro RLBSL_LKSB = 20,0,32,0 %;
1786 0 macro RLBSW_STATUS = 20,0,16,0 %;
1787 0 macro RLBSW_S_BITS = 22,0,16,0 %;
1788 0 macro RLBSL_LOCK_ID = 24,0,32,0 %;
1789 0
1790 0 !*** MODULE $FLBDEF ***
1791 0
1792 0 ! file lock block definitions
1793 0
1794 0 literal FLB$C_BID = 23;
1795 0 literal FLB$K_BLN = 20;
1796 0 literal FLB$C_BLN = 20;
1797 0 literal FLB$$FLBDEF = 20;
1798 0 macro FLBSL_FLB_LNK = 0,0,32,0 %;
1799 0 macro FLBSL_RLB_LNK = 4,0,32,0 %;
1800 0 macro FLBSB_BID = 8,0,8,0 %;
1801 0 macro FLBSB_BLN = 9,0,8,0 %;
1802 0 macro FLBSL_IFB_PTR = 12,0,32,0 %;
1803 0 macro FLBSL_LOCK_ID = 16,0,32,0 %;
1804 0
1805 0 !*** MODULE $SRCDEF ***
1806 0
1807 0 ! directory cache node definitions

```

```

! length of rlb
! length of rlb
! link to next rlb
! longword definition to optimize clearing field
! more flag bits
! Timer queued.
! 3'rd word of records rfa
! id for iox f.o.
! block id
! block length in longwords
! propagation of ROP TMO field
! various locking flags
! propagation of ROP WAT bit
! defines lock manager mode "concurrent read"
! allow reader access to locked record flag
! used to query lock database
! defines lock manager option "convert"
! sets lock as "level 2" RU consistency
! this RLB contains no lock.
! propagation of ROP TMO bit
! 1'st and 2'nd words of record's rfa
! identification of owning stream
! first longword of lock status block
! VMS status code
! various status bits
! second longword of lkbs is lock_id

```

```

1808 0 |
1809 0 | literal DRC$K_BLN = 62; | length of directory cache node
1810 0 | literal DRC$C_BLN = 62; | length of directory cache node
1811 0 | literal DRC$$DRCDEF = 62;
1812 0 | macro DRC$NXTFLNK = 0,0,32,0 %; | link to next entry, this level
1813 0 | macro DRC$NXTBLNK = 4,0,32,0 %; | link to previous entry, this level
1814 0 | macro DRC$LVFLNK = 8,0,32,0 %; | link to first entry, next lower level
1815 0 | macro DRC$LVLBLNK = 12,0,32,0 %; | link to last entry, next lower level
1816 0 | ! note: the links are maintained in lru order
1817 0 | macro DRC$T_NAME = 16,0,0,0 %;
1818 0 | literal DRC$$NAME = 40; | directory name or device and unit
1819 0 | ! note: stored as counted string counting count itself
1820 0 | macro DRC$W_DID = 56,0,0,0 %;
1821 0 | literal DRC$$DID = 6; | file id for directory
1822 0 | macro DRC$W_DIRSEQ = 58,0,16,0 %; | directory sequence ! for device node
1823 0 |
1824 0 | *** MODULE $RLSDEF ***
1825 0 |
1826 0 | release option flag definitions
1827 0 |
1828 0 | literal RLSSM_RETURN = 1;
1829 0 | literal RLSSM_WRT_THRU = 2;
1830 0 | literal RLSSM_KEEP_LOCK = 4;
1831 0 | literal RLSSM_DEQ = 8;
1832 0 | literal RLSS$RLSDEF = 1;
1833 0 | macro RLSSV_RETURN = 0,0,1,0 %; | return buffer and bdb to free space lists
1834 0 | macro RLSSV_WRT_THRU = 0,1,1,0 %; | write buffer if dirty
1835 0 | macro RLSSV_KEEP_LOCK = 0,2,1,0 %; | keep bdb locked
1836 0 | macro RLSSV_DEQ = 0,3,1,0 %; | always release lock
1837 0 |
1838 0 | *** MODULE $CSHDEF ***
1839 0 |
1840 0 | cache option flag definitions
1841 0 |
1842 0 | literal CSHSM_LOCK = 1;
1843 0 | literal CSHSM_NOWAIT = 2;
1844 0 | literal CSHSM_NOREAD = 4;
1845 0 | literal CSHSM_NOBUFFER = 8;
1846 0 | literal CSH$CSHDEF = 1;
1847 0 | macro CSHSV_LOCK = 0,0,1,0 %; | obtain exclusive access to block
1848 0 | macro CSHSV_NOWAIT = 0,1,1,0 %; | do not wait for block on access interlock
1849 0 | macro CSHSV_NOREAD = 0,2,1,0 %; | do not read in block
1850 0 | macro CSHSV_NOBUFFER = 0,3,1,0 %; | obtain access to block but don't allocate
1851 0 | ! a buffer for it and don't read it
1852 0 | ! collision
1853 0 |
1854 0 | *** MODULE $PIODEF ***
1855 0 |
1856 0 | rms overall status bit definitions
1857 0 |
1858 0 | literal PIO$$PIODEF = 1;
1859 0 | macro PIO$V_IHAST = 0,0,1,0 %; | set if asts implicitly inhibited
1860 0 | ! if reset by disabled ast, ast must be re-
1861 0 | ! enabled
1862 0 | macro PIO$V_EOD = 0,1,1,0 %; | set if searching for 'eod' string on 'input'
1863 0 | macro PIO$V_SYNC1 = 0,2,1,0 %; | sync stalled operation using efn 27
1864 0 |

```

```

1865 0 macro PIO$V_SYNC2 = 0,3,1,0 %;      ! sync stalled operation using efn 28
1866 0
1867 0 *** MODULE $FTLDEF ***
1868 0
1869 0      definitions for rms debug failure codes
1870 0
1871 0
1872 0      the following codes are for temporary bug check tests, and are
1873 0      internal to rms.  all of the codes are negative, implying that they
1874 0      do not return to the caller, probably killing the process (if not
1875 0      the entire system).
1876 0
1877 0      literal FTL$ SETPRTFAIL = -1;      ! set protection system service failed (rm0bufmgr)
1878 0      literal FTL$ STKTOOBIG = -2;      ! stack too big for asb (rm0stall)
1879 0      literal FTL$ BADIFAB = -3;      ! invalid ifab or irab (rm0fset,rm0conn,rm0rset,rm0prflnm)
1880 0      literal FTL$ GTCHNFAIL = -4;      ! get channel system service failure (rm0prflnm)
1881 0      literal FTL$ BADORGCASE = -5;      ! invalid orgcase value for dispatch (all rms$
1882 0      ! level routines except open and create)
1883 0      literal FTL$ BADBDB = -6;      ! block not a bdb (rm0bufmgr)
1884 0      literal FTL$ ASBALLFAIL = -7;      ! couldn't allocate an asb (rm0stall)
1885 0      literal FTL$ BADASTPRM = -8;      ! ast parameter not a valid ifab/irab addr (rm0stall)
1886 0      literal FTL$ CANTDOAST = -9;      ! couldn't redeclare ast (insf. mem.) (rm0stall)
1887 0      literal FTL$ NOSTRUCT = -10;      ! rab or fab not same on ast (rm0stall)
1888 0      literal FTL$ NOASB = -11;      ! asb not allocated or stream not busy on ast (rm0stall)
1889 0      literal FTL$ NONXTBDB = -12;      ! no next bdb available (rm1seqxfr)
1890 0      literal FTL$ BADBUFSIZ = -13;      ! disk buffer size not = 512 (rm1conn)
1891 0      literal FTL$ ENQDEQFAIL = -14;      ! enq or deq service failed (rm0reclck)
1892 0      literal FTL$ NOCURBDB = -15;      ! no current bdb before calling rm$release (rm0reclck)
1893 0      literal FTL$ NOPARENT = -16;      ! no parent lock available for global buffer section lock (rm0share)
1894 0      literal FTL$ DEALLERR = -17;      ! ifab deallocation attempted with other block(s)
1895 0      ! still allocated (rms0close)
1896 0      literal FTL$ IORNDN = -18;      ! i/o rundown inconsistency (either ifab or irab
1897 0      ! table entries not zeroed) (rms0rndwn)
1898 0      literal FTL$ XFERSIZE = -19;      ! size of requested transfer not equal to
1899 0      ! or less than the current number of bytes
1900 0      ! in use for the bdb (rm0cache)
1901 0      literal FTL$ NOTLOCKED = -20;      ! bdb not locked and a keep lock request
1902 0      ! was made on a release request.
1903 0      literal FTL$ NODIDORFID = -21;      ! neither a fid nor a did was set upon exit from
1904 0      ! rm$setdid (rms0erase)
1905 0      literal FTL$ RELEASFAIL = -22;      ! release of non-dirty bdb failed (rm0xtnd23,rms0extend)
1906 0      literal FTL$ NOLOCKBDB = -23;      ! no lock bdb found (rm0xtnd23)
1907 0      literal FTL$ NONETWORK = -24;      ! network routine entered but no network support in rms
1908 0      literal FTL$ LOCKFAILED = -25;      ! failed to lock prolog (rm2create)
1909 0      literal FTL$ BADLEVEL = -26;      ! to search by id, structure level must be 0
1910 0      literal FTL$ ASTDECERR = -27;      ! ast declaration for file sharing failed
1911 0      literal FTL$ BADGBLCNT = -28;      ! Zero global buffer count found when not expected (rm1ccnn)
1912 0      literal FTL$ ACCNTOVFLO = -29;      ! access count overflow (rm0share)
1913 0      literal FTL$ BDBAVAIL = -30;      ! BDB was available and shouldn't have been.
1914 0      literal FTL$ GBLNOLK = -31;      ! Record locking was not set with global buffers.
1915 0      literal FTL$ LCKFND = -32;      ! A lock was found and we don't know what to do.
1916 0      literal FTL$ NOBLB = -33;      ! No BLB was found and there should have been one.
1917 0      literal FTL$ NOGBPGB = -34;      ! No GBPGB was found and should have been.
1918 0      literal FTL$ NOLCLBUF = -35;      ! Should have found a local buffer.
1919 0      literal FTL$ NORDNOTSET = -36;      ! NOREAD not set when NOBUFFER was.
1920 0      literal FTL$ NOTGBPGB = -37;      ! Found an illegit BDB.
1921 0      literal FTL$ NOSFSB = -38;      ! No SFSB when allocating BLB.

```



```

1922 0 literal FTL$_LOCKHELD = -39;      ! Attempted to return a BLB with lock_id neq 0
1923 0 literal FTL$_RLSDRT = -40;      ! Dirty buffer found in releasall.
1924 0 literal FTL$_BADBLB = -41;      ! Bad BLB found in blocking AST routine.
1925 0 literal FTL$_BADOWNER = -42;    ! Owner field in BLB is bad in blocking AST routine.
1926 0 literal FTL$_GETLKIFAIL = -43;  ! $GETLKIW failed in last chance (rms0lstch).
1927 0 literal FTL$_BADEBKHBK = -44;  ! tried to store an invalid EBK/HBK (rm0share).
1928 0
1929 0 *** MODULE $BUGDEF ***
1930 0
1931 0     the following internal codes are for non-fatal bug check reporting.
1932 0     these codes are positive byte values. they trigger a reporting action
1933 0     and return to the caller with r0 set to rms$_bug+<8*the bug code>,
1934 0     which is an externally documented rms error code.
1935 0
1936 0 literal BUG$_BADDFLTDIR = 1;      ! DEFAULT DIRECTORY STRING INVALID (RMOXPFN)
1937 0
1938 0 *** MODULE $IDXDEF ***
1939 0
1940 0     IDX field definitions
1941 0
1942 0     index descriptor definition
1943 0
1944 0     An index descriptor block exists for each key of reference in use.
1945 0     they are not necessarily contiguous in memory.
1946 0
1947 0 literal IDX$_BID = 15;              ! id for index descriptor block
1948 0 literal IDX$_DUPKEYS = 1;
1949 0 literal IDX$_CHGKEYS = 2;
1950 0 literal IDX$_NULKEYS = 4;
1951 0 literal IDX$_IDX COMPR = 8;
1952 0 literal IDX$_INITIDX = 16;
1953 0 literal IDX$_KEY COMPR = 64;
1954 0 literal IDX$_REC COMPR = 128;
1955 0 literal IDX$_STRING = 0;          ! string data type
1956 0 literal IDX$_SGNWORD = 1;         ! signed binary word
1957 0 literal IDX$_UNSGNWORD = 2;      ! unsigned binary word
1958 0 literal IDX$_SGNLONG = 3;        ! signed binary, long word
1959 0 literal IDX$_UNSGNLONG = 4;      ! unsigned binary long word
1960 0 literal IDX$_PACKED = 5;         ! packed decimal
1961 0 literal IDX$_SGNQWAD = 6;        ! signed binary quadword
1962 0 literal IDX$_UNSGNQWAD = 7;      ! unsigned binary quadword
1963 0 literal IDX$_V2 BKT = 0;         ! Prologue two bucket
1964 0 literal IDX$_CMPIDX = 1;         ! Prologue 3, index keys are compressed
1965 0 literal IDX$_NCMPIDX = 2;        ! Prologue 3, index keys are not compressed
1966 0 literal IDX$_CMPCMP = 3;        ! Prologue 3, primary key is compressed, data
1967 0     is compressed
1968 0     Prologue 3, SDR key is compressed
1969 0 literal IDX$_CMPNCMP = 4;        ! Prologue 3, primary key is compressed,
1970 0     data is not compressed
1971 0 literal IDX$_NCMPCMP = 5;        ! Prologue 3, primary key is not compressed
1972 0     data is compressed
1973 0 literal IDX$_NCMPNCMP = 6;      ! Prologue 3, primary key is not compressed
1974 0     data is not compressed
1975 0     Prologue 3, SDR key is compressed
1976 0 literal IDX$_FIXED_BLN = 44;
1977 0 literal IDX$_FIXED_BLN = 44;
1978 0

```

the following is the length of the fixed part of the index descriptor

the following is repeated for each key segment

literal	IDX\$S_IDXDEF = 48;	
macro	IDX\$S_IDXFL = 0,0,32,0 %;	forward link to next index descriptor
macro	IDX\$B_BID = 8,0,8,0 %;	block id
macro	IDX\$B_BLN = 9,0,8,0 %;	length of block
macro	IDX\$S_VBN = 16,0,32,0 %;	VBN where the descriptor came from
macro	IDX\$S_OFFSET = 14,0,16,0 %;	Offset into the block (VBN) of the descriptor
macro	IDX\$B_DESC_NO = 16,0,8,0 %;	Descriptor number (index into update buffer)
macro	IDX\$B_IANUM = 18,0,8,0 %;	area number for index buckets
macro	IDX\$B_LANUM = 19,0,8,0 %;	area number for lower index buckets
macro	IDX\$B_DANUM = 20,0,8,0 %;	area number for data buckets
macro	IDX\$B_ROOTLEV = 21,0,8,0 %;	level of root
macro	IDX\$B_IDXBKTSZ = 22,0,8,0 %;	size of index bucket in vbn's
macro	IDX\$B_DATBKTSZ = 23,0,8,0 %;	size of data bucket in vbn's
macro	IDX\$S_ROOTVBN = 24,0,32,0 %;	start vbn of root bucket
macro	IDX\$B_FLAGS = 28,0,8,0 %;	index/key flags
macro	IDX\$S_DUPKEYS = 28,0,1,0 %;	duplicate keys allowed
macro	IDX\$S_CHGKEYS = 28,1,1,0 %;	keys can change values
macro	IDX\$S_NULKEYS = 28,2,1,0 %;	null key value allowed
macro	IDX\$S_IDX_COMPR = 28,3,1,0 %;	index is compressed
macro	IDX\$S_INITIDX = 28,4,1,0 %;	index is not initialized
macro	IDX\$S_KEY_COMPR = 28,6,1,0 %;	key has been compressed
macro	IDX\$S_REC_COMPR = 28,7,1,0 %;	data record is in compressed form
macro	IDX\$B_DATATYPE = 29,0,8,0 %;	data type of key field
macro	IDX\$B_SEGMENTS = 30,0,8,0 %;	number of key field segments
macro	IDX\$B_NULLCHAR = 31,0,8,0 %;	null character
macro	IDX\$B_KEYSZ = 32,0,8,0 %;	total key size
macro	IDX\$B_KEYREF = 33,0,8,0 %;	key of reference(0-primary)
macro	IDX\$S_MINRECSZ = 34,0,16,0 %;	minimum record size
macro	IDX\$S_IDXFILL = 36,0,16,0 %;	index fill
macro	IDX\$S_DATFILL = 38,0,16,0 %;	data fill
macro	IDX\$B_IDXBKTTYP = 40,0,8,0 %;	PLG3 - type of index bucket and SDR bucket
macro	IDX\$B_DATBKTTYP = 41,0,8,0 %;	PLG3 - type of primary data bucket
macro	IDX\$S_POSITION = 44,0,16,0 %;	key segment position
macro	IDX\$B_SIZE = 46,0,8,0 %;	key segment size (plg 3)
macro	IDX\$B_TYPE = 47,0,8,0 %;	key segment datatype (plg 3)

*** MODULE \$UPDDEF ***

update buffer flags

literal UPD\$M_INS_NEW = 1;
 literal UPD\$M_OLD_DEL = 2;
 literal UPD\$S_UPDDEF = 1;
 macro UPD\$B_FLAGS = 0,0,8,0 %;
 macro UPD\$S_INS_NEW = 0,0,1,0 %;
 macro UPD\$S_OLD_DEL = 0,1,1,0 %;

! alternate key to be inserted from record buffer
 ! delete this key value using old record

*** MODULE \$GBHDEF ***

GBH field definitions

Global Buffer Header (GBH)

There is a Global Buffer Header for every file's global buffer section.

*** WARNING - THIS STRUCTURE MUST BE QUADWORD ALIGNED ***

```

literal GBH$C_BID = 17;          ! Block ID code for GBH
literal GBH$M_CACHE_IN = 1;
literal GBH$M_CACHE_OUT = 2;
literal GBH$M_RLS_IN = 4;
literal GBH$M_RLS_OUT = 8;
literal GBH$M_QIO_START = 16;
literal GBH$M_QIO_DONE = 32;
literal GBH$M_STALL = 64;
literal GBH$M_THREADGO = 128;
literal GBH$M_BLB_ENQ = 256;
literal GBH$M_BLB_GRANT = 512;
literal GBH$M_BLB_DEQ = 1024;
literal GBH$M_BLB_BLOCK = 2048;
literal GBH$M_F1 = 4096;
literal GBH$M_F2 = 8192;
literal GBH$M_F3 = 16384;
literal GBH$M_F4 = 32768;
literal GBH$K_BLN = 88;          ! Length of global buffer header structure
literal GBH$C_BLN = 88;          ! Length of global buffer header structure
literal GBH$S_GBHDEF = 88;
macro GBH$L_GBD_FLNK = 0,0,32,0 %; ! Self relative queue header for GBD's
macro GBH$L_GBD_BLNK = 4,0,32,0 %;
macro GBH$B_BID = 8,0,8,0 %;      ! Block ID
macro GBH$B_BLN = 9,0,8,0 %;      ! Length of GBH in longwords
macro GBH$V_TRC_FLGS = 10,0,16,0 %; ! Trace flags (set to trace given function)
macro GBH$V_CACHE_IN = 10,0,1,0 %; ! Cache inputs
macro GBH$V_CACHE_OUT = 10,1,1,0 %; ! Cache outputs
macro GBH$V_RLS_IN = 10,2,1,0 %;   ! Release inputs
macro GBH$V_RLS_OUT = 10,3,1,0 %;  ! Release outputs
macro GBH$V_QIO_START = 10,4,1,0 %; ! Qio inputs
macro GBH$V_QIO_DONE = 10,5,1,0 %; ! Qio outputs
macro GBH$V_STALL = 10,6,1,0 %;     ! Stall inputs
macro GBH$V_THREADGO = 10,7,1,0 %;  ! Stall outputs
macro GBH$V_BLB_ENQ = 10,8,1,0 %;   ! Bucket lock ENQ inputs
macro GBH$V_BLB_GRANT = 10,9,1,0 %; ! Bucket lock grant status
macro GBH$V_BLB_DEQ = 10,10,1,0 %;  ! Bucket lock DEQ request
macro GBH$V_BLB_BLOCK = 10,11,1,0 %; ! Blocking AST received
macro GBH$V_F1 = 10,12,1,0 %;
macro GBH$V_F2 = 10,13,1,0 %;
macro GBH$V_F3 = 10,14,1,0 %;
macro GBH$V_F4 = 10,15,1,0 %;
macro GBH$L_HI_VBN = 12,0,32,0 %;   ! Highest possible VBN value (FFFFFFFF).
macro GBH$L_GS_SIZE = 16,0,32,0 %;  ! Size of total section in bytes.
macro GBH$L_LOCK_ID = 20,0,32,0 %;  ! Lock ID of system file lock.
macro GBH$L_GS_LOCK_ID = 24,0,32,0 %; ! Lock ID of system global section lock.
macro GBH$L_USECNT = 28,0,32,0 %;    ! Accessor count for section.
macro GBH$L_TRC_FLNK = 32,0,32,0 %;  ! Trace blocks forward link
macro GBH$L_TRC_BLNK = 36,0,32,0 %;  ! Trace blocks back link
macro GBH$L_GBD_START = 40,0,32,0 %; ! Offset to first GBD.
macro GBH$L_GBD_END = 44,0,32,0 %;   ! Offset to last GBD.
macro GBH$L_GBD_NEXT = 48,0,32,0 %;  ! Offset to next cache victim GBD.
macro GBH$L_SCAN_NUM = 52,0,32,0 %;  ! Number of GBD's to scan for victim.

```

2093 0
 2094 0
 2095 0
 2096 0
 2097 0
 2098 0
 2099 0
 2100 0
 2101 0
 2102 0
 2103 0
 2104 0
 2105 0
 2106 0
 2107 0
 2108 0
 2109 0
 2110 0
 2111 0
 2112 0
 2113 0
 2114 0
 2115 0
 2116 0
 2117 0
 2118 0
 2119 0
 2120 0
 2121 0
 2122 0
 2123 0
 2124 0
 2125 0
 2126 0
 2127 0
 2128 0
 2129 0
 2130 0
 2131 0
 2132 0
 2133 0
 2134 0
 2135 0
 2136 0
 2137 0
 2138 0
 2139 0
 2140 0
 2141 0
 2142 0
 2143 0
 2144 0
 2145 0
 2146 0
 2147 0
 2148 0
 2149 0

Global buffer statistics section

macro GBH\$H_HIT = 56,0,32,0 %; Buffer found in global cache
 macro GBH\$H_MISS = 60,0,32,0 %; Buffer not found in global cache
 macro GBH\$H_READ = 64,0,32,0 %; Buffer read from disk into cache
 macro GBH\$H_WRITE = 68,0,32,0 %; Buffer written from cache to disk
 macro GBH\$H_DFW_WRITE = 72,0,32,0 %; Deferred writeback from cache to disk
 macro GBH\$H_CROSS_HIT = 76,0,32,0 %; Cross process hit count.
 macro GBH\$H_OUTBUFQUO = 80,0,32,0 %; Count of times GBLBUFQUO limit was hit.
 macro GBH\$H_FILL_1 = 84,0,32,0 %; Force quadword alignment

*** MODULE \$TRCDEF ***

TRC field definitions

Trace block structure (TRC)

Tracing saves at specific points in the RMS code for debugging and algorithm analysis purposes.

*** WARNING - THIS STRUCTURE MUST BE QUADWORD ALIGNED ***

literal TRC\$C_BID = 18; Trace block code
 literal TRC\$K_BLN = 64; NOTE: should be quadwords multiple to
 literal TRC\$C_BLN = 64; NOTE: should be quadwords multiple to
 literal TRC\$S_TRCDEF = 64;
 macro TRC\$H_FLNK = 0,0,32,0 %; Trace block forward link
 macro TRC\$H_BLNK = 4,0,32,0 %; Trace block back link
 macro TRC\$B_BID = 8,0,8,0 %; Block ID
 macro TRC\$B_BLN = 9,0,8,0 %; Length of block in longwords
 macro TRC\$W_FUNCTION = 10,0,16,0 %; Function code (see GBH definitions)
 macro TRC\$H_STRUCTURE = 12,0,32,0 %; Ifab/irab address.
 macro TRC\$W_PID = 16,0,16,0 %; Process ID
 macro TRC\$W_SEQNUM = 18,0,16,0 %; Sequence number.
 macro TRC\$H_VBN = 20,0,32,0 %; VBN requested.
 macro TRC\$H_RETURN1 = 24,0,32,0 %; Address of caller.
 macro TRC\$H_RETURN2 = 28,0,32,0 %; Caller's caller.
 macro TRC\$H_ARGS = 32,0,0,0 %;
 literal TRC\$S_ARGS = 32; Function specific arguments
 macro TRC\$H_ARG_FLG = 32,0,32,0 %; Argument flags (R3).
 macro TRC\$H_BDB_ADDR = 36,0,32,0 %; BDB address.
 macro TRC\$W_BDB_USERS = 40,0,16,0 %; Use count from BDB.
 macro TRC\$W_BDB_BUFF = 42,0,16,0 %; BDB buffer ID.
 macro TRC\$B_BDB_CACHE = 44,0,8,0 %; BDB cache value.
 macro TRC\$B_BDB_FLAGS = 45,0,8,0 %; Status flags from BDB.
 macro TRC\$H_BDB_SEQ = 46,0,32,0 %; Sequence number from BDB.
 macro TRC\$B_BLB_MODE = 50,0,8,0 %; Mode held in BLB.
 macro TRC\$B_BLB_FLAGS = 51,0,8,0 %; Flags from BLB.
 macro TRC\$H_BLB_ADDR = 52,0,32,0 %; Address of BLB.
 macro TRC\$H_BLB_LOCK = 56,0,32,0 %; Lock ID from BLB.
 macro TRC\$H_BLB_SEQ = 60,0,32,0 %; Sequence number from BLB.

! maintain quad alignment on header

*** MODULE \$GBDDEF ***

GBD structure definitions

Global Buffer Descriptor (GBD)

There is a single GBD for every buffer in a global buffer section (used only with shared files). The GBD's themselves are in the section also and linked from a queue header in the Global Buffer Header (GBH).

*** WARNING - THIS STRUCTURE MUST BE QUADWORD ALIGNED ***

```

literal GBD$C_BID = 19;          ! Block ID code for GBD
literal GBD$M_VALID = 1;
literal GBD$K_BLN = 40;          ! Length of Global Buffer Descriptor structure.
literal GBD$C_BLN = 40;          ! Length of Global Buffer Descriptor structure.
literal GBD$S_GBDDEF = 40;
macro GBD$L_FLNK = 0,0,32,0 %;    ! Forward link - Note: This is a self relative queue
macro GBD$L_BLNK = 4,0,32,0 %;    ! Back link
macro GBD$B_BID = 8,0,8,0 %;      ! Block ID
macro GBD$B_BLN = 9,0,8,0 %;      ! Block length of GBD
macro GBD$B_FLAGS = 10,0,8,0 %;   ! Buffer status flags
macro GBD$V_VALID = 10,0,1,0 %;   ! Buffer is valid.
macro GBD$B_CACHE_VAL = 11,0,8,0 %; ! Cache value of this bucket
macro GBD$L_VBN = 12,0,32,0 %;     ! VBN of bucket the buffer describes
macro GBD$L_VBNSEQNUM = 16,0,32,0 %; ! VBN sequence number validity check
macro GBD$L_LOCK_ID = 20,0,32,0 %; ! Lock ID of system lock.
macro GBD$W_NUMB = 24,0,16,0 %;    ! Number of bytes in use
macro GBD$W_SIZE = 26,0,16,0 %;    ! Size of buffer in bytes
macro GBD$L_REL_ADDR = 28,0,32,0 %; ! Address of buffer relative to GBH
macro GBD$W_USECNT = 32,0,16,0 %;  ! Accessor count for bucket
macro GBD$B_REHIT_RD = 34,0,8,0 %; ! Rehit by same process count.
macro GBD$B_REHIT_LK = 35,0,8,0 %; ! Rehit by same locker process.

```

*** MODULE \$BLBDEF ***

BLB field definitions

Bucket Lock Block (BLB)

The BLB contains the argument list for the SYS\$ENQ system service as well a pointer to the BDB it relates to and other status.

```

literal BLB$C_BID = 16;          ! BLB code
literal BLB$M_LOCK = 1;
literal BLB$M_NOWAIT = 2;
literal BLB$M_NOREAD = 4;
literal BLB$M_NOBUFFER = 8;
literal BLB$M_IOLOCK = 16;
literal BLB$M_DFW = 32;
literal BLB$M_WRITEBACK = 64;
literal BLB$K_BLN = 56;          ! Length of BLB
literal BLB$C_BLN = 56;          ! Length of BLB
literal BLB$S_BLBDEF = 56;
macro BLB$L_FLNK = 0,0,32,0 %;    ! Link to next BLB
macro BLB$L_BLNK = 4,0,32,0 %;    ! Back link
macro BLB$B_PID = 8,0,8,0 %;      ! Block ID
macro BLB$B_BLN = 9,0,8,0 %;      ! Block length
macro BLB$B_BLBFLGS = 10,0,8,0 %; ! Control flags for BLB

```

```

2207 0 macro BLBSV_LOCK = 10,0,1,0 %;
2208 0 macro BLBSV_NOWAIT = 10,1,1,0 %;
2209 0 macro BLBSV_NOREAD = 10,2,1,0 %;
2210 0 macro BLBSV_NOBUFFER = 10,3,1,0 %;
2211 0 macro BLBSV_IOLOCK = 10,4,1,0 %;
2212 0 macro BLBSV_DFW = 10,5,1,0 %;
2213 0 macro BLBSV_WRITEBACK = 10,6,1,0 %;
2214 0 macro BLBSB_MODEHELD = 11,0,8,0 %;
2215 0 macro BLBSL_BDB_ADDR = 12,0,32,0 %;
2216 0 macro BLBSL_OWNER = 16,0,32,0 %;
2217 0 macro BLBSL_VBN = 20,0,32,0 %;
2218 0 macro BLBSL_RESDSC = 24,0,0,0 %;
2219 0 literal BLBS RESDSC = 8;
2220 0 macro BLBSW_LRSTS = 32,0,16,0 %;
2221 0 macro BLBSL_LOCK_ID = 36,0,32,0 %;
2222 0 macro BLBSL_VALBLK = 40,0,0,0 %;
2223 0 literal BLBS VALBLK = 16;
2224 0 macro BLBSL_VALSEQNO = 40,0,32,0 %;
2225 0
2226 0 !*** MODULE $RJBDEF ***
2227 0
2228 0     RJB Definitions
2229 0
2230 0     RMS Journaling Block (RJB)
2231 0
2232 0     This block contains the necessary control information to keep
2233 0     track of the state of journaling on this file
2234 0
2235 0 literal RJB$C_BID = 22;
2236 0 literal RJB$K_BLN = 12;
2237 0 literal RJB$C_BLN = 12;
2238 0 literal RJB$M_RU = 1;
2239 0 literal RJB$M_BI = 2;
2240 0 literal RJB$M_AI = 4;
2241 0 literal RJB$M_AT = 8;
2242 0 literal RJB$M_OPEN = 16;
2243 0 literal RJB$S_RJBDEF = 12;
2244 0 macro RJB$Q_CHAN = 0,0,0,0 %;
2245 0 literal RJB$S_CHAN = 8;
2246 0 macro RJB$W_ROCHAN = 0,0,16,0 %;
2247 0 macro RJB$W_BICHAN = 2,0,16,0 %;
2248 0 macro RJB$W_AICHAN = 4,0,16,0 %;
2249 0 macro RJB$W_ATCHAN = 6,0,16,0 %;
2250 0 macro RJB$B_BID = 8,0,8,0 %;
2251 0 macro RJB$B_BLN = 9,0,8,0 %;
2252 0 macro RJB$W_FLAGS = 10,0,16,0 %;
2253 0 macro RJB$V_RU = 10,0,1,0 %;
2254 0 macro RJB$V_BI = 10,1,1,0 %;
2255 0 macro RJB$V_AI = 10,2,1,0 %;
2256 0 macro RJB$V_AT = 10,3,1,0 %;
2257 0 macro RJB$V_OPEN = 10,4,1,0 %;
2258 0
2259 0 !*** MODULE $MJBDEF ***
2260 0
2261 0     MJB field definitions
2262 0
2263 0     Miscellaneous Journaling Buffer

```

```

! Corresponds to CSH$V_LOCK
! Same as CSH$V_NOWAIT
! Same as CSH$V_NOREAD
! Same as CSH$V_NOBUFFER
! Lock mode for read/write
! This is lock for deferred write buffer
! The associated buffer must be written back
! Mode of current lock held.
! BDB for which this lock is held
! Address of stream owning this lock
! VBN of bucket lock (resource name)
!
! Resource name descriptor
! Lock status word
! Lock ID
!
! Lock value block
! Sequence number part of value block
!
! Length of RJB
! Length of RJB
!
! Channel Block
! channel for recovery unit journal
! channel for before image journal
! channel for after image journal
! channel for audit trail journal
! Block Id
! Block Length
! Flags word
! Set to indicate RU channel open
! Set to indicate BI channel open
! Set to indicate AI channel open
! Set to indicate AT channel open
! Indicates $OPEN mapping entry written

```

```

2264 0
2265 0
2266 0
2267 0
2268 0
2269 0
2270 0
2271 0
2272 0
2273 0
2274 0
2275 0
2276 0
2277 0
2278 0
2279 0
2280 0
2281 0
2282 0
2283 0
2284 0
2285 0
2286 0
2287 0
2288 0
2289 0
2290 0
2291 0
2292 0
2293 0
2294 0
2295 0
2296 0
2297 0
2298 0
2299 0
2300 0
2301 0
2302 0
2303 0
2304 0
2305 0
2306 0
2307 0
2308 0
2309 0
2310 0
2311 0
2312 0
2313 0
2314 0
2315 0
2316 0
2317 0
2318 0
2319 0
2320 0

```

The MJB is used for writing miscellaneous journal entries,
for example, extend entries or audit-trail entries.

```

literal MJB$C_BID = 24;
literal MJB$M_INIT = 1;
literal MJB$M_FORCE = 1;
literal MJB$M_FILE = 4;
literal MJB$M_SYNCH_SHARE = 8;
literal MJB$K_BLN = 32;
literal MJB$C_BLN = 32;
literal MJB$S_MJBDEF = 32;
macro MJB$B_BID = 8,0,8,0 %;
macro MJB$B_BLN = 9,0,8,0 %;
macro MJB$W_FLAGS = 10,0,16,0 %;
macro MJB$V_INIT = 10,0,1,0 %;
macro MJB$V_FORCE = 10,1,1,0 %;
macro MJB$V_FILE = 10,2,1,0 %;
macro MJB$V_SYNCH_SHARE = 10,3,1,0 %;
! STALL
! and not buffered by CJF (input to WRITE_MJB)
macro MJB$B_JNL = 12,0,8,0 %;
macro MJB$Q_DESC = 16,0,0,0 %;
literal MJB$S_DESC = 8;
macro MJB$W_SIZE = 16,0,16,0 %;
macro MJB$L_POINTER = 20,0,32,0 %;
macro MJB$Q_IOSB = 24,0,0,0 %;
literal MJB$S_IOSB = 8;
macro MJB$T_RJR = 32,0,0,0 %;

```

block id
block length in longwords
flags
set if RJR overhead is initialized
set if RJR is to be written thru to journal
set if file operation to journal
set if file lock can't be released during

```

*****
* Copyright (c) 1982, 1983
* by DIGITAL Equipment Corporation, Maynard, Mass.
*
* This software is furnished under a license and may be used and copied
* only in accordance with the terms of such license and with the
* inclusion of the above copyright notice. This software or any other
* copies thereof may not be provided or otherwise made available to any
* other person. No title to and ownership of the software is hereby
* transferred.
*
* The information in this software is subject to change without notice
* and should not be construed as a commitment by DIGITAL Equipment
* Corporation.
*
* DIGITAL assumes no responsibility for the use or reliability of its
* software on equipment which is not supplied by DIGITAL.
*****
Created 15-SEP-1984 22:54:20 by VAX-11 SDL V2.0 Source: 15-SEP-1984 22:49:17 _S255$DUA28:[RMS.SRC]RMSFWAD
*****
*** MODULE $FWADEF ***
**

```

2321 0
2322 0
2323 0
2324 0
2325 0
2326 0
2327 0
2328 0
2329 0
2330 0
2331 0
2332 0
2333 0
2334 0
2335 0
2336 0
2337 0
2338 0
2339 0
2340 0
2341 0
2342 0
2343 0
2344 0
2345 0
2346 0
2347 0
2348 0
2349 0
2350 0
2351 0
2352 0
2353 0
2354 0
2355 0
2356 0
2357 0
2358 0
2359 0
2360 0
2361 0
2362 0
2363 0
2364 0
2365 0
2366 0
2367 0
2368 0
2369 0
2370 0
2371 0
2372 0
2373 0
2374 0
2375 0
2376 0
2377 0

Flags

```
--
literal FWASM_DUPOK = 1;
literal FWASM_NOCOPY = 2;
literal FWASM_SL_PASS = 4;
literal FWASM_RLF_PASS = 8;
literal FWASM_FNA_PASS = 16;
literal FWASM_NAM_DVI = 32;
literal FWASM_EXP_NODE = 64;
literal FWASM_VERSION = 2048;
literal FWASM_TYPE = 4096;
literal FWASM_NAME = 8192;
literal FWASM_DIR = 16384;
literal FWASM_DEVICE = 32768;
literal FWASM_EXP_VER = 65536;
literal FWASM_EXP_TYPE = 131072;
literal FWASM_EXP_NAME = 262144;
literal FWASM_WC_VER = 524288;
literal FWASM_WC_TYPE = 1048576;
literal FWASM_WC_NAME = 2097152;
literal FWASM_EXP_DIR = 4194304;
literal FWASM_EXP_DEV = 8388608;
literal FWASM_WILDCARD = 16777216;
literal FWASM_NODE = 33554432;
literal FWASM_QUOTED = 67108864;
literal FWASM_GRPMBR = 134217728;
literal FWASM_WILD_DIR = 268435456;
literal FWASM_DIR_VLS = -536870912;
literal FWASC_ALL = 248;
```

constant for all flags that vary per parsing pass

```
literal FWASC_ALLPASS = 25;
++
```

Misc. Fields

```
--
literal FWASC_BID = 40;          ! bid of fwa
literal FWASC_MAXNODNAM = 6;     ! max node name size
literal FWASC_MAXLNDDNAM = 15;   ! max logical node name size
literal FWASC_MAXNODLST = 127;   ! max node spec list size (concatenated node specs)
```

device name descriptor

```
literal FWASC_MAXDEVICE = 255;   ! max device name size
literal FWASC_MAXCDIR = 8;       ! max number of concealed directories
literal FWASC_MAXSUBDIR = 7;     ! max number of sub directories
literal FWASC_MAXDIRLEN = 255;   ! max size of total directory spec
```

```
should be: top + subdir  39 * 8
            dots between  7
            delimiters    2
            -----
            total         321
```



```

2378 0      The filename, filetype and fileversion descriptors MUST be contiguous
2379 0
2380 0      file name descriptor
2381 0
2382 0      literal FWASC_MAXNAME = 39;          ! max file name size
2383 0      literal FWASC_MAXQUOTED = 255;        ! max quoted string size
2384 0
2385 0      file type descriptor
2386 0
2387 0      literal FWASC_MAXTYPE = 39;          ! max file type size
2388 0
2389 0      file version number descriptor
2390 0
2391 0      literal FWASC_MAXVER = 6;             ! maximum version
2392 0      literal FWASC_MAXRNS = 86;            ! max resultant name string size
2393 0      literal FWASC_STATBLK = 10;           ! define length of statistics block
2394 0      literal FWASC_BLN_FWA = 500;          ! length of fwa
2395 0      literal FWASC_BLN_FWA = 500;          ! length of fwa
2396 0      literal FWASC_MAXSUBNOD = 7;         ! max number of secondary (sub) node specs
2397 0      ++
2398 0
2399 0      buffers for parsed filename elements
2400 0
2401 0      --
2402 0      literal FWASC_FIBLEN = 76;            ! fib buffer size
2403 0      literal FWASC_DIRBUFSIZ = 39;         ! size of each directory buffer
2404 0
2405 0      rooted directory name buffers
2406 0
2407 0      NOTE: These buffers must be contiguous
2408 0
2409 0      literal FWASC_BLN_BUF = 2364;          ! length of fwa and buffers
2410 0      literal FWASC_BLN_BUF = 2364;          ! length of fwa and buffers
2411 0      literal FWASC_BLN_BUF = 2364;          ! length of fwa and buffers
2412 0      literal FWASC_BLN_BUF = 2364;          ! length of fwa and buffers
2413 0      literal FWASC_FWADEF = 2364;
2414 0      macro FWASC_FLAGS = 0,0,0,0 %;
2415 0      literal FWASC_FLAGS = 8;               ! various parse status flags
2416 0      macro FWASC_PASSFLGS = 0,0,8,0 %;      ! flags for pass only
2417 0      macro FWASC_FLDFLGS = 1,0,8,0 %;        ! flags for fields seen
2418 0      macro FWASC_WILDFLGS = 2,0,8,0 %;        ! flags for wild cards
2419 0      macro FWASC_PARSEFLGS = 3,0,8,0 %;       ! flags for parse results
2420 0      macro FWASC_DIRFLGS = 4,0,8,0 %;        ! flags primarily for directory spec
2421 0      macro FWASC_DIRWCFGLGS = 5,0,8,0 %;      ! directory wild flags
2422 0      macro FWASC_LNFLGS = 6,0,8,0 %;         ! logical name flag byte
2423 0      macro FWASC_SLFLGS = 7,0,8,0 %;         ! search list + rooted directory flags
2424 0
2425 0      flags for pass
2426 0
2427 0      macro FWASC_DUPOK = 0,0,1,0 %;          ! discard duplicate element
2428 0      macro FWASC_NOCOPY = 0,1,1,0 %;         ! do not copy this field
2429 0      macro FWASC_SL_PASS = 0,2,1,0 %;        ! search list pass
2430 0      macro FWASC_RLF_PASS = 0,3,1,0 %;       ! set if applying related file defaults
2431 0      macro FWASC_FNA_PASS = 0,4,1,0 %;       ! set if primary name string parse pass
2432 0      macro FWASC_NAM_DVI = 0,5,1,0 %;        ! set if open by name block
2433 0      macro FWASC_EXP_NODE = 0,6,1,0 %;       ! explicit node has been seen, null or normal
2434 0

```

```

2435 0      | flags for fields seen
2436 0
2437 0      | macro FWASV_VERSION = 0,11,1,0 %;      | set if version seen
2438 0      | macro FWASV_TYPE = 0,12,1,0 %;          | set if type seen
2439 0      | macro FWASV_NAME = 0,13,1,0 %;          | set if name seen
2440 0      | macro FWASV_DIR = 0,14,1,0 %;           | set if directory spec seen
2441 0      | macro FWASV_DEVICE = 0,15,1,0 %;        | set if device seen
2442 0
2443 0      | flags for wild cards
2444 0
2445 0      | macro FWASV_EXP_VER = 0,16,1,0 %;        | set if explicit version
2446 0      | macro FWASV_EXP_TYPE = 0,17,1,0 %;       | set if explicit type
2447 0      | macro FWASV_EXP_NAME = 0,18,1,0 %;       | set if explicit name
2448 0      | macro FWASV_WC_VER = 0,19,1,0 %;         | set if wildcard (*) version
2449 0      | macro FWASV_WC_TYPE = 0,20,1,0 %;        | " type
2450 0      | macro FWASV_WC_NAME = 0,21,1,0 %;        | " name
2451 0      | macro FWASV_EXP_DIR = 0,22,1,0 %;        | set if explicit directory
2452 0      | macro FWASV_EXP_DEV = 0,23,1,0 %;        | set if explicit device
2453 0
2454 0      | flags for parse results
2455 0
2456 0      | macro FWASV_WILDCARD = 0,24,1,0 %;       | set if any wildcard seen
2457 0      | macro FWASV_NODE = 0,25,1,0 %;           | set if node name seen
2458 0      | macro FWASV_QUOTED = 0,26,1,0 %;         | set if quoted string seen
2459 0      | macro FWASV_GRPMBR = 0,27,1,0 %;         | set if directory in [grp,mbr] format
2460 0      | macro FWASV_WILD_DIR = 0,28,1,0 %;       | inclusive or of directory wild cards
2461 0      | macro FWASV_DIR_LVL = 0,29,3,0 %;        |
2462 0      | literal FWASV_DIR_LVL = 3;               | ! of directory sublevels (0 = ufd only)
2463 0      | (valid only if node set and no fldflgs)
2464 0
2465 0      | flags primarily for directory spec
2466 0
2467 0      | macro FWASV_DIR1 = 4,0,1,0 %;            | ufd level directory or group seen
2468 0      | macro FWASV_DIR2 = 4,1,1,0 %;            | sfd level 1 directory or member seen
2469 0
2470 0      | directory wild flags
2471 0
2472 0      | macro FWASV_WILD_UFD = 4,8,1,0 %;         | the dir1 spec was a wild card
2473 0      | macro FWASV_WILD_SFD1 = 4,9,1,0 %;       | the dir2 spec was a wild card
2474 0      | macro FWASV_WILD_GRP = 4,8,1,0 %;        | the grp spec contained a wild card
2475 0      | macro FWASV_WILD_MBR = 4,9,1,0 %;        | the mbr spec contained a wild card
2476 0
2477 0      | logical name flag and miscellaneous byte
2478 0
2479 0      | macro FWASV_LOGNAME = 4,16,1,0 %;        | a logical name has been seen this pass
2480 0      | (note: this byte is saved as context
2481 0      | when processing [.dir-list] format)
2482 0      | macro FWASV_OBJTYPE = 4,17,1,0 %;        | set if quoted string is of the
2483 0      | "objectType=..." form
2484 0      | (valid only if quoted set)
2485 0      | macro FWASV_NETSTR = 4,18,1,0 %;         | set if quoted string is of the
2486 0      | "objectType=taskname/..." form
2487 0      | (valid only if quoted and obdtype set)
2488 0      | macro FWASV_DEV_UNDER = 4,19,1,0 %;      | device name was prefixed with an underscore
2489 0      | macro FWASV_FILEFOUND = 4,20,1,0 %;      | true if at least one file found by search
2490 0      | macro FWASV_REMRESULT = 4,21,1,0 %;      | use resultant string returned by fal
2491 0      | macro FWASV_SYNTAX_CHK = 4,22,1,0 %;     | syntax-only checking is requested (NAMSV SYNCHK set)

```

```

2492 0
2493 0 search list and rooted directory flag byte
2494 0
2495 0 macro FWASV_SLPRESENT = 4,24,1,0 %; search list present
2496 0 macro FWASV_CONCEAL_DEV = 4,25,1,0 %; concealed device present
2497 0 macro FWASV_ROOT_DIR = 4,26,1,0 %; root directory present
2498 0 macro FWASV_DFLT_MFD = 4,27,1,0 %; default MFD string inserted, due to [-]
2499 0 macro FWASV_EXP_ROOT = 4,28,1,0 %; explicit root directory
2500 0
2501 0 Value for all filename elements except node
2502 0
2503 0 macro FWASB_BID = 8,0,8,0 %; bid
2504 0 macro FWASB_BLN = 9,0,8,0 %; bln
2505 0 macro FWASB_DIRTERM = 10,0,8,0 %; directory spec terminator (']' or '>')
2506 0 macro FWASB_ROOTERM = 11,0,8,0 %; root directory spec terminator (']' or '>')
2507 0 macro FWASL_ESCSTRING = 12,0,32,0 %; escape equivalence string
2508 0 macro FWASB_ESCFLG = 12,0,8,0 %; set to the char <esc> if an escape string
2509 0 ! seen, zero otherwise
2510 0 macro FWASB_ESCTYP = 13,0,8,0 %; escape 'type' byte
2511 0 macro FWASW_ESCIFI = 14,0,16,0 %; escape ifi value
2512 0 macro FWASQ_FIB = 16,0,0,0 %;
2513 0 literal FWASS_FIB = 8; fib descriptor
2514 0 macro FWASL_DEVBUSIZ = 24,0,32,0 %; device buffer size
2515 0 macro FWASL_DEV_CLASS = 28,0,32,0 %; device class
2516 0 macro FWASL_RECSIZ = 32,0,32,0 %; blocked record size
2517 0 macro FWASL_UNIT = 36,0,32,0 %; device unit number
2518 0 macro FWASL_UIC = 40,0,32,0 %; file owner uic
2519 0 macro FWASW_PRO = 44,0,16,0 %; file protection word
2520 0 macro FWASB_DIRLEN = 46,0,8,0 %; overall directory spec length
2521 0 macro FWASB_SUBNODCNT = 47,0,8,0 %; number of secondary (sub) node specs found
2522 0 macro FWASL_DIRBDB = 48,0,32,0 %; address of directory file bdb
2523 0 macro FWASL_LOOKUP = 52,0,32,0 %; address of new directory cache node
2524 0 macro FWASL_DEVNODADR = 56,0,32,0 %; address of device directory cache node
2525 0 macro FWASQ_DIR = 60,0,0,0 %;
2526 0 literal FWASS_DIR = 8; directory name scratch buffer
2527 0 macro FWASL_UCHAR = 68,0,32,0 %; user characteristics longword
2528 0 macro FWASW_UCHAR = 68,0,16,0 %;
2529 0 macro FWASL_FWA_PTR = 72,0,32,0 %; pointer to second fwa if any ($RENAME)
2530 0 macro FWASL_SWB_PTR = 76,0,32,0 %; pointer to swb
2531 0 macro FWASL_BUF_PTR = 80,0,32,0 %; address of temporary buffer
2532 0 macro FWASL_IMPURE_AREA = 84,0,32,0 %; saved R11 (rm$xpfn only)
2533 0 macro FWASL_ATR_WORK = 88,0,32,0 %; pointer to work area for ACP attributes
2534 0 (zero one not currently allocated)
2535 0 ++
2536 0
2537 0 Logical name and search list fields
2538 0
2539 0 --
2540 0
2541 0 Item list block for logical name services
2542 0
2543 0 macro FFAST_ITMLST = 92,0,0,0 %;
2544 0 literal FWASS_ITMLST = 64; logical name item list
2545 0 macro FFAST_ITM_INDEX = 92,0,0,0 %;
2546 0 literal FWASS_ITM_INDEX = 12; index
2547 0 macro FFAST_ITM_ATTR = 104,0,0,0 %;
2548 0 literal FWASS_ITM_ATTR = 12; attributes

```

```

2549 0 macro FWAST_ITM_STRING = 116,0,0,0 %;
2550 0 literal FWASS_ITM_STRING = 12; ! string
2551 0 macro FWAST_ITM_MAX_INDEX = 128,0,0,0 %;
2552 0 literal FWASS_ITM_MAX_INDEX = 12; ! max index
2553 0 macro FWASL_ITM_END = 140,0,32,0 %; ! terminating longword
2554 0
2555 0 Logical name translation fields
2556 0
2557 0 macro FWASB_BUFFLG = 156,0,8,0 %; ! flag for which translation buffer is in use
2558 0 (0 = buf2 in use, -1 = buf1 in use)
2559 0 macro FWASB_XLTMODE = 157,0,8,0 %; ! mode of translation on input to $TRNLNM
2560 0 mode of equivalence string on output from $TRNLNM
2561 0 macro FWASW_XLTSIZ = 158,0,16,0 %; ! length of equivalence string
2562 0 macro FWASL_XLTBUFF1 = 160,0,32,0 %; ! primary translation buffer descriptor
2563 0 macro FWASL_XLTBUFF2 = 164,0,32,0 %; ! secondary translation buffer descriptor
2564 0
2565 0 SLBH and SLB pointers
2566 0
2567 0 macro FWASL_SLBH_PTR = 168,0,32,0 %; ! current SLB list
2568 0 macro FWASL_SLB_PTR = 172,0,32,0 %; ! current SLB list
2569 0 macro FWASL_SLBH_FLINK = 176,0,32,0 %; ! SLBH que fwd link
2570 0 macro FWASL_SLBH_BLINK = 180,0,32,0 %; ! SLBH que back link
2571 0
2572 0 Fake SLB - NOTE: This MUST be the size of SLB$C_BLN
2573 0 The field FWASB_LEVEL must be at the same offset
2574 0 as SLB$Q_LEVEL would be. (It sounds like a real
2575 0 hack but it works very nicely)
2576 0
2577 0 macro FWAST_SLB = 184,0,0,0 %;
2578 0 literal FWASS_SLB = 24; ! space for SLB$C_BLN
2579 0 macro FWASB_LEVEL = 195,0,8,0 %; ! recursion level
2580 0
2581 0 Logical name descriptor
2582 0
2583 0 macro FWASQ_LOGNAM = 208,0,0,0 %;
2584 0 literal FWASS_LOGNAM = 8; ! logical name descriptor
2585 0 ++
2586 0
2587 0 descriptors for parsed filename elements
2588 0
2589 0
2590 0 The descriptors are defined as:
2591 0
2592 0 -----
2593 0 flags | length
2594 0 -----
2595 0 address
2596 0 -----
2597 0
2598 0
2599 0 The flags are defined by FSCB$V_flag in $FSCBDEF
2600 0
2601 0 --
2602 0 macro FWASQ_NODE = 216,0,0,0 %;
2603 0 literal FWASS_NODE = 8; ! node name (actually node spec list) descriptor
2604 0 (the associated buffer is fwa$t_nodebuf)
2605 0 macro FWASQ_DEVICE = 224,0,0,0 %;

```

```

2606 0 literal FWASS_DEVICE = 8; ! device name descriptor
2607 0 macro FWASQ_CONCEAL_DEV = 232,0,0,0 %;
2608 0 literal FWASS_CONCEAL_DEV = 8; ! concealed device descriptor
2609 0
2610 0 directory name descriptors NOTE: The two sets of directory
2611 0 descriptors must be contiguous
2612 0 or RM$SETDID will break
2613 0
2614 0 macro FWASQ_CDIR1 = 240,0,0,0 %;
2615 0 literal FWASS_CDIR1 = 8; ! concealed top directory descriptors
2616 0 macro FWASQ_CDIR2 = 248,0,0,0 %;
2617 0 literal FWASS_CDIR2 = 8; ! concealed subdirectory 1
2618 0 macro FWASQ_CDIR3 = 256,0,0,0 %;
2619 0 literal FWASS_CDIR3 = 8; ! " " 2
2620 0 macro FWASQ_CDIR4 = 264,0,0,0 %;
2621 0 literal FWASS_CDIR4 = 8; ! " " 3
2622 0 macro FWASQ_CDIR5 = 272,0,0,0 %;
2623 0 literal FWASS_CDIR5 = 8; ! " " 4
2624 0 macro FWASQ_CDIR6 = 280,0,0,0 %;
2625 0 literal FWASS_CDIR6 = 8; ! " " 5
2626 0 macro FWASQ_CDIR7 = 288,0,0,0 %;
2627 0 literal FWASS_CDIR7 = 8; ! " " 6
2628 0 macro FWASQ_CDIR8 = 296,0,0,0 %;
2629 0 literal FWASS_CDIR8 = 8; ! " " 7
2630 0 macro FWASQ_DIR1 = 304,0,0,0 %;
2631 0 literal FWASS_DIR1 = 8; ! top level directory descriptors
2632 0 macro FWASQ_DIR2 = 312,0,0,0 %;
2633 0 literal FWASS_DIR2 = 8; ! subdirectory 1
2634 0 macro FWASQ_DIR3 = 320,0,0,0 %;
2635 0 literal FWASS_DIR3 = 8; ! " 2
2636 0 macro FWASQ_DIR4 = 328,0,0,0 %;
2637 0 literal FWASS_DIR4 = 8; ! " 3
2638 0 macro FWASQ_DIR5 = 336,0,0,0 %;
2639 0 literal FWASS_DIR5 = 8; ! " 4
2640 0 macro FWASQ_DIR6 = 344,0,0,0 %;
2641 0 literal FWASS_DIR6 = 8; ! " 5
2642 0 macro FWASQ_DIR7 = 352,0,0,0 %;
2643 0 literal FWASS_DIR7 = 8; ! " 6
2644 0 macro FWASQ_DIR8 = 360,0,0,0 %;
2645 0 literal FWASS_DIR8 = 8; ! " 7
2646 0 macro FWASQ_NAME = 368,0,0,0 %;
2647 0 literal FWASS_NAME = 8; ! file name descriptor
2648 0 macro FWASQ_QUOTED = 368,0,0,0 %;
2649 0 literal FWASS_QUOTED = 8; ! quoted string descriptor
2650 0 macro FWASQ_TYPE = 376,0,0,0 %;
2651 0 literal FWASS_TYPE = 8; ! file type descriptor
2652 0 macro FWASQ_VERSION = 384,0,0,0 %;
2653 0 literal FWASS_VERSION = 8; ! file version descriptor
2654 0 macro FWASQ_RNS = 392,0,0,0 %;
2655 0 literal FWASS_RNS = 8; ! resultant name string descriptor
2656 0 macro FWASQ_SHRFIL = 400,0,0,0 %;
2657 0 literal FWASS_SHRFIL = 8; ! shared file device descriptor (readable form)
2658 0 macro FWASQ_SHRFIL_LCK = 408,0,0,0 %;
2659 0 literal FWASS_SHRFIL_LCK = 8; ! shared file device descriptor (unreadable form - used for lock name)
2660 0 macro FWASQ_AS_SHRFIL = 416,0,0,0 %;
2661 0 literal FWASS_AS_SHRFIL = 8; ! secondary device descriptor (readable form)
2662 0 macro FWAST_STATBLK = 424,0,0,0 %;

```

```

2663 0 literal FWASS_STATBLK = 10;
2664 0 macro FWASL_SBN = 424,0,32,0 %;
2665 0 macro FWASL_HBK = 428,0,32,0 %;
2666 0
2667 0 node descriptors
2668 0
2669 0 macro FWASQ_NODE1 = 436,0,0,0 %;
2670 0 literal FWASS_NODE1 = 8;
2671 0 ! (the associated buffer is fwaSt_nodebuf)
2672 0 macro FWASQ_NODE2 = 444,0,0,0 %;
2673 0 literal FWASS_NODE2 = 8;
2674 0 macro FWASQ_NODE3 = 452,0,0,0 %;
2675 0 literal FWASS_NODE3 = 8;
2676 0 macro FWASQ_NODE4 = 460,0,0,0 %;
2677 0 literal FWASS_NODE4 = 8;
2678 0 macro FWASQ_NODE5 = 468,0,0,0 %;
2679 0 literal FWASS_NODE5 = 8;
2680 0 macro FWASQ_NODE6 = 476,0,0,0 %;
2681 0 literal FWASS_NODE6 = 8;
2682 0 macro FWASQ_NODE7 = 484,0,0,0 %;
2683 0 literal FWASS_NODE7 = 8;
2684 0 macro FWASQ_NODE8 = 492,0,0,0 %;
2685 0 literal FWASS_NODE8 = 8;
2686 0 macro FFAST_FIBBUF = 500,0,0,0 %;
2687 0 literal FWASS_FIBBUF = 76;
2688 0 macro FFAST_RNM_FID = 576,0,0,0 %;
2689 0 literal FWASS_RNM_FID = 6;
2690 0
2691 0 directory name buffers
2692 0
2693 0 NOTE: These buffers must be contiguous
2694 0
2695 0 macro FFAST_DIR1BUF = 582,0,0,0 %;
2696 0 literal FWASS_DIR1BUF = 39;
2697 0 macro FFAST_DIR2BUF = 621,0,0,0 %;
2698 0 literal FWASS_DIR2BUF = 39;
2699 0 macro FFAST_DIR3BUF = 660,0,0,0 %;
2700 0 literal FWASS_DIR3BUF = 39;
2701 0 macro FFAST_DIR4BUF = 699,0,0,0 %;
2702 0 literal FWASS_DIR4BUF = 39;
2703 0 macro FFAST_DIR5BUF = 738,0,0,0 %;
2704 0 literal FWASS_DIR5BUF = 39;
2705 0 macro FFAST_DIR6BUF = 777,0,0,0 %;
2706 0 literal FWASS_DIR6BUF = 39;
2707 0 macro FFAST_DIR7BUF = 816,0,0,0 %;
2708 0 literal FWASS_DIR7BUF = 39;
2709 0 macro FFAST_DIR8BUF = 855,0,0,0 %;
2710 0 literal FWASS_DIR8BUF = 39;
2711 0 macro FFAST_CDIR1BUF = 894,0,0,0 %;
2712 0 literal FWASS_CDIR1BUF = 39;
2713 0 macro FFAST_CDIR2BUF = 933,0,0,0 %;
2714 0 literal FWASS_CDIR2BUF = 39;
2715 0 macro FFAST_CDIR3BUF = 972,0,0,0 %;
2716 0 literal FWASS_CDIR3BUF = 39;
2717 0 macro FFAST_CDIR4BUF = 1011,0,0,0 %;
2718 0 literal FWASS_CDIR4BUF = 39;
2719 0 macro FFAST_CDIR5BUF = 1050,0,0,0 %;

```

```

! starting lbn if contiguous
! high vbn

```

```

! primary node spec descriptor

```

```

! secondary (sub) node spec descriptors (1-7)

```

```

! note: bytes 2-3 of each of these descriptors

```

```

! contains the flags word that is output

```

```

! from nextfld subroutine in rm0xpfn

```

```

! note: fwaSq_node1 thru 'fwaSq_node8'

```

```

! describe the same string as does

```

```

! fwaSq_node

```

```

! fib buffer

```

```

! saved fid for rename directory check

```

```

! ufd level (or group)

```

```

! 1st sfd level (or member)

```

```

! subdirectory 2

```

```

! subdirectory 3

```

```

! subdirectory 4

```

```

! subdirectory 5

```

```

! subdirectory 6

```

```

! subdirectory 7

```

```

! ufd level (or group)

```

```

! 1st sfd level (or member)

```

```

! subdirectory 2

```

```

! subdirectory 3

```

```

2720 0 literal FWASS CDIR5BUF = 39; ! subdirectory 4
2721 0 macro FFAST CDIR6BUF = 1089,0,0,0 %;
2722 0 literal FWASS CDIR6BUF = 39; ! subdirectory 5
2723 0 macro FFAST CDIR7BUF = 1128,0,0,0 %;
2724 0 literal FWASS CDIR7BUF = 39; ! subdirectory 6
2725 0 macro FFAST CDIR8BUF = 1167,0,0,0 %;
2726 0 literal FWASS CDIR8BUF = 39; ! subdirectory 7
2727 0
2728 0 NOTES: 1. The following buffers must be contiguous as eventually the
2729 0 type and version are appended to the name string
2730 0
2731 0 2. The name buffer and the type buffer must be 1 byte larger then
2732 0 the max name and type size (resp) because xpfm writes the
2733 0 name and type terminators in the buffer at the end of the string
2734 0
2735 0 macro FFAST NAMEBUF = 1206,0,0,0 %;
2736 0 literal FWASS NAMEBUF = 256; ! file name/quoted string buffer
2737 0 macro FFAST TYPEBUF = 1462,0,0,0 %;
2738 0 literal FWASS TYPEBUF = 40; ! file type buffer
2739 0 macro FFAST VERBUF = 1502,0,0,0 %;
2740 0 literal FWASS VERBUF = 6; ! file version buffer
2741 0 macro FFAST UCBSLSTS = 1508,0,32,0 %; ! ucb$sl_sts field for prim device
2742 0 macro FFAST UNDER DEV = 1512,0,8,0 %; ! character "_" stored here
2743 0 macro FFAST DEVICEBUF = 1513,0,0,0 %;
2744 0 literal FWASS DEVICEBUF = 255; ! device name buffer
2745 0 macro FFAST CDEVICEBUF = 1768,0,0,0 %;
2746 0 literal FWASS CDEVICEBUF = 256; ! concealed device name buffer
2747 0 macro FFAST UNDER NOD = 2024,0,8,0 %; ! character "_" stored here
2748 0 macro FFAST NODEBUF = 2025,0,0,0 %;
2749 0 literal FWASS NODEBUF = 127; ! node name buffer
2750 0 macro FFAST WILD = 2152,0,0,0 %;
2751 0 literal FWASS WILD = 48; ! scratch field used by RMOWILD
2752 0 size = count 1
2753 0 name 39
2754 0 .dir:* 6
2755 0 spare 2
2756 0 -----
2757 0 48
2758 0 macro FFAST SHRFILBUF = 2200,0,0,0 %;
2759 0 literal FWASS SHRFILBUF = 16; ! shared file device id buffer (readable form)
2760 0 macro FFAST SHRFIL LCKNAM = 2216,0,0,0 %;
2761 0 literal FWASS SHRFIL LCKNAM = 16; ! shared file device id buffer (unreadable form - used for lock name)
2762 0 macro FFAST AS SHRFILBUF = 2232,0,0,0 %;
2763 0 literal FWASS AS SHRFILBUF = 16; ! secondary device id buffer (readable form)
2764 0 macro FFAST BIJNL = 2248,0,0,0 %;
2765 0 literal FWASS BIJNL = 8; ! descriptor of BI journal name
2766 0 macro FFAST AIJNL = 2256,0,0,0 %;
2767 0 literal FWASS AIJNL = 8; ! descriptor of AI journal name
2768 0 macro FFAST ATJNL = 2264,0,0,0 %;
2769 0 literal FWASS ATJNL = 8; ! descriptor of AT journal name
2770 0 macro FFAST BIACE = 2272,0,0,0 %;
2771 0 literal FWASS BIACE = 20; ! BI journal name ACE
2772 0 macro FFAST BIJNLN = 2276,0,0,0 %;
2773 0 literal FWASS BIJNLN = 16;
2774 0 macro FFAST AIACE = 2292,0,0,0 %;
2775 0 literal FWASS AIACE = 20; ! AI journal name ACE
2776 0 macro FFAST AIJNLN = 2296,0,0,0 %;

```

```

2777 0 literal FWASS AIJNLN = 16;
2778 0 macro FFAST ATACE = 2312,0,0,0 %;
2779 0 literal FWASS ATACE = 20; ! AT journal name ACE
2780 0 macro FFAST ATJNLN = 2316,0,0,0 %;
2781 0 literal FWASS ATJNLN = 16;
2782 0 macro FFAST IDACE = 2332,0,0,0 %;
2783 0 literal FWASS IDACE = 32; ! Journal ID ACE
2784 0 macro FFAST JNLID = 2336,0,0,0 %;
2785 0 literal FWASS JNLID = 28; ! complete journal ID
2786 0 macro FFAST VOLNAM = 2336,0,0,0 %;
2787 0 literal FWASS VOLNAM = 12; ! volume lable of media file resides on
2788 0 macro FFAST FID = 2348,0,0,0 %;
2789 0 literal FWASS FID = 6; ! file-id
2790 0 macro FFAST ID_DATE = 2356,0,0,0 %;
2791 0 literal FWASS_ID_DATE = 8; ! id time stamp
2792 0
2793 0 !*** MODULE $SLBHDEF ***
2794 0
2795 0 SLBH - Search List Header Block
2796 0
2797 0 literal SLBH$C_BID = 43; ! ID
2798 0 literal SLBH$K_BLN = 20; ! length of SLBH
2799 0 literal SLBH$C_BLN = 20; ! length of SLBH
2800 0 literal SLBH$S_SLBHDEF = 20;
2801 0 macro SLBH$S_FLINK = 0,0,32,0 %; ! forward link
2802 0 macro SLBH$S_BLINK = 4,0,32,0 %; ! backward link
2803 0 macro SLBH$B_BID = 8,0,8,0 %; ! block ID
2804 0 macro SLBH$B_BLN = 9,0,8,0 %; ! length
2805 0 macro SLBH$B_PASSFLGS = 10,0,8,0 %; ! flags for FWA$B_PASSFLGS
2806 0 macro SLBH$B_STR_LEN = 11,0,8,0 %; ! string length
2807 0 macro SLBH$S_SLB_QUEUE = 12,0,32,0 %; ! ptr to SLB queue
2808 0 macro SLBH$S_NAM_FNB = 16,0,32,0 %; ! saved FNB from RLF file
2809 0 macro SLBH$S_STRING = 20,0,0,0 %; ! start of string
2810 0
2811 0 !*** MODULE $SLBDEF ***
2812 0
2813 0 SLB - Search List Block
2814 0
2815 0 literal SLB$C_BID = 41; ! ID
2816 0 literal SLB$M_REALSLB = 1;
2817 0 literal SLB$K_BLN = 24; ! length of SLB
2818 0 literal SLB$C_BLN = 24; ! length of SLB
2819 0 literal SLB$S_SLBDEF = 24;
2820 0 macro SLB$S_FLINK = 0,0,32,0 %; ! forward link
2821 0 macro SLB$S_BLINK = 4,0,32,0 %; ! backward link
2822 0 macro SLB$B_BID = 8,0,8,0 %; ! block ID
2823 0 macro SLB$B_BLN = 9,0,8,0 %; ! length
2824 0 macro SLB$B_FLAGS = 10,0,8,0 %; ! flags
2825 0 macro SLB$V_REALSLB = 10,0,1,0 %; ! "Real" SLB as opposed to the fake FWA one
2826 0 macro SLB$B_LEVEL = 11,0,8,0 %; ! recursion level
2827 0 macro SLB$S_INDEX = 12,0,32,0 %; ! translation index
2828 0 macro SLB$S_MAX_INDEX = 16,0,32,0 %; ! max translation index
2829 0 macro SLB$S_ATTR = 20,0,32,0 %; ! attributes flags
2830 0
2831 0 !*** MODULE $FSCBDEF ***
2832 0
2833 0 ! FSCB - FileScan control block

```


This block is passed to PARSE_STRING from XPFN and RMS\$FILESCAN

The descriptors are defined as:



descriptor flags

These flags are used through out the RMS file name parsing routines.
The flags can be found in all of the field descriptors.

NOTE: The flag ELIPS must be the first bit in the second word.
It is referenced this way in RMOWILD and other places

literal FSCBSM_ELIPS = 65536;
literal FSCBSM_WILD = 131072;
literal FSCBSM_ACS = 262144;
literal FSCBSM_QUOTED = 524288;
literal FSCBSM_NULL = 1048576;
literal FSCBSM_PWD = 2097152;
literal FSCBSM_GRPMBR = 4194304;
literal FSCBSM_MINUS = 8388608;
literal FSCBSM_CONCEAL = 16777216;
literal FSCBSM_MFD = 33554432;
literal FSCBSM_ROOTED = 67108864;
literal FSCBSM_FSCBDEF = 4;
macro FSCBSV_ELIPS = 0,16,1,0 %;
macro FSCBSV_WILD = 0,17,1,0 %;
macro FSCBSV_ACS = 0,18,1,0 %;
macro FSCBSV_QUOTED = 0,19,1,0 %;
macro FSCBSV_NULL = 0,20,1,0 %;
macro FSCBSV_PWD = 0,21,1,0 %;
macro FSCBSV_GRPMBR = 0,22,1,0 %;
macro FSCBSV_MINUS = 0,23,1,0 %;
macro FSCBSV_CONCEAL = 0,24,1,0 %;
macro FSCBSV_MFD = 0,25,1,0 %;
macro FSCBSV_ROOTED = 0,26,1,0 %;

: elipssis was detected in directory (dir)
: a wild card was detected (dir,name,type,ver)
: access control string in node name (node)
: quoted file spec (name)
: field was null (terminator only) (all)
: password masked out (set in xpfm) (node)
: group,member format directory (dir)
: minus directory field (dir)
: name was concealed (dev)
: MFD directory (set in xpfm) (dir)
: directory was a root directory (dir)

FSCB

literal FSCBSM_NODE = 1;
literal FSCBSM_DEVICE = 2;
literal FSCBSM_ROOT = 4;
literal FSCBSM_DIRECTORY = 8;
literal FSCBSM_NAME = 16;
literal FSCBSM_TYPE = 32;
literal FSCBSM_VERSION = 64;
literal FSCBSM_MAXNODE = 8;
literal FSCBSM_MAXROOT = 8;

: max number of node descriptors
: max number of root descriptors

```

2891 0 literal FSCB$K_BLN = 260;
2892 0 literal FSCB$C_BLN = 260;
2893 0 literal FSCB$C_MAXDIR = 8;      ! max number of directory descriptors
2894 0 literal FSCB$S_FSCBDEF1 = 260;
2895 0 macro FSCB$B_FCD_FLAGS = 0,0,8,0 %;      ! field flags
2896 0 macro FSCB$V_NODE = 0,0,1,0 %;
2897 0 macro FSCB$V_DEVICE = 0,1,1,0 %;
2898 0 macro FSCB$V_ROOT = 0,2,1,0 %;
2899 0 macro FSCB$V_DIRECTORY = 0,3,1,0 %;
2900 0 macro FSCB$V_NAME = 0,4,1,0 %;
2901 0 macro FSCB$V_TYPE = 0,5,1,0 %;
2902 0 macro FSCB$V_VERSION = 0,6,1,0 %;
2903 0 macro FSCB$B_NODES = 1,0,8,0 %;      ! number of nodes in spec
2904 0 macro FSCB$B_ROOTS = 2,0,8,0 %;      ! number of root directories in spec
2905 0 macro FSCB$B_DIRS = 3,0,8,0 %;      ! number of directories in spec
2906 0 macro FSCB$Q_FILESPEC = 4,0,0,0 %;
2907 0 literal FSCB$S_FILESPEC = 8;      ! full file spec
2908 0 macro FSCB$Q_NODE = 12,0,0,0 %;
2909 0 literal FSCB$S_NODE = 8;      ! full node list spec
2910 0 macro FSCB$Q_DEVICE = 20,0,0,0 %;
2911 0 literal FSCB$S_DEVICE = 8;      ! device spec
2912 0 macro FSCB$Q_ROOT = 28,0,0,0 %;
2913 0 literal FSCB$S_ROOT = 8;      ! full root directory list spec
2914 0 macro FSCB$Q_DIRECTORY = 36,0,0,0 %;
2915 0 literal FSCB$S_DIRECTORY = 8;      ! full directory list spec
2916 0 macro FSCB$Q_NAME = 44,0,0,0 %;
2917 0 literal FSCB$S_NAME = 8;      ! file name
2918 0 macro FSCB$Q_TYPE = 52,0,0,0 %;
2919 0 literal FSCB$S_TYPE = 8;      ! file type
2920 0 macro FSCB$Q_VERSION = 60,0,0,0 %;
2921 0 literal FSCB$S_VERSION = 8;      ! file version
2922 0 macro FSCB$Q_NODE1 = 68,0,0,0 %;
2923 0 literal FSCB$S_NODE1 = 8;      ! the NODEn descriptors must be contiguous
2924 0 macro FSCB$Q_NODE2 = 76,0,0,0 %;
2925 0 literal FSCB$S_NODE2 = 8;
2926 0 macro FSCB$Q_NODE3 = 84,0,0,0 %;
2927 0 literal FSCB$S_NODE3 = 8;
2928 0 macro FSCB$Q_NODE4 = 92,0,0,0 %;
2929 0 literal FSCB$S_NODE4 = 8;
2930 0 macro FSCB$Q_NODE5 = 100,0,0,0 %;
2931 0 literal FSCB$S_NODE5 = 8;
2932 0 macro FSCB$Q_NODE6 = 108,0,0,0 %;
2933 0 literal FSCB$S_NODE6 = 8;
2934 0 macro FSCB$Q_NODE7 = 116,0,0,0 %;
2935 0 literal FSCB$S_NODE7 = 8;
2936 0 macro FSCB$Q_NODE8 = 124,0,0,0 %;
2937 0 literal FSCB$S_NODE8 = 8;
2938 0 macro FSCB$Q_ROOT1 = 132,0,0,0 %;
2939 0 literal FSCB$S_ROOT1 = 8;      ! the ROOTn descriptors must be contiguous
2940 0 macro FSCB$Q_ROOT2 = 140,0,0,0 %;
2941 0 literal FSCB$S_ROOT2 = 8;
2942 0 macro FSCB$Q_ROOT3 = 148,0,0,0 %;
2943 0 literal FSCB$S_ROOT3 = 8;
2944 0 macro FSCB$Q_ROOT4 = 156,0,0,0 %;
2945 0 literal FSCB$S_ROOT4 = 8;
2946 0 macro FSCB$Q_ROOT5 = 164,0,0,0 %;
2947 0 literal FSCB$S_ROOT5 = 8;

```

```

2948 0 macro FSCBSQ ROOT6 = 172,0,0,0 %;
2949 0 literal FSCBSS ROOT6 = 8;
2950 0 macro FSCBSQ ROOT7 = 180,0,0,0 %;
2951 0 literal FSCBSS ROOT7 = 8;
2952 0 macro FSCBSQ ROOT8 = 188,0,0,0 %;
2953 0 literal FSCBSS ROOT8 = 8;
2954 0 macro FSCBSQ DIRECTORY1 = 196,0,0,0 %;
2955 0 literal FSCBSS DIRECTORY1 = 8;
2956 0 macro FSCBSQ DIRECTORY2 = 204,0,0,0 %;
2957 0 literal FSCBSS DIRECTORY2 = 8;
2958 0 macro FSCBSQ DIRECTORY3 = 212,0,0,0 %;
2959 0 literal FSCBSS DIRECTORY3 = 8;
2960 0 macro FSCBSQ DIRECTORY4 = 220,0,0,0 %;
2961 0 literal FSCBSS DIRECTORY4 = 8;
2962 0 macro FSCBSQ DIRECTORY5 = 228,0,0,0 %;
2963 0 literal FSCBSS DIRECTORY5 = 8;
2964 0 macro FSCBSQ DIRECTORY6 = 236,0,0,0 %;
2965 0 literal FSCBSS DIRECTORY6 = 8;
2966 0 macro FSCBSQ DIRECTORY7 = 244,0,0,0 %;
2967 0 literal FSCBSS DIRECTORY7 = 8;
2968 0 macro FSCBSQ DIRECTORY8 = 252,0,0,0 %;
2969 0 literal FSCBSS_DIRECTORY8 = 8;
2970 0
2971 0 :*** MODULE $SWBDEF ***
2972 0
2973 0 : Directory string work buffer for wild card directory processing
2974 0
2975 0 literal SWBSM_ELLIPSIS = 1;
2976 0 literal SWBSM_BOUNDED = 2;
2977 0 literal SWBSM_WILD = 4;
2978 0 literal SWBSM_DELIMITER = 8;
2979 0 literal SWBSM_TRAVERSE = 16;
2980 0 literal SWBSM_FIRST = 32;
2981 0 literal SWBSM_ELLIPSIS_EXISTS = 64;
2982 0 literal SWBSM_VALID_DID = 128;
2983 0 literal SWBSC_BID = 42;
2984 0 literal SWBSK_BLN = 328;
2985 0 literal SWBSC_BLN = 328;
2986 0 : wild dir spec
2987 0 literal SWBSS_SWBDEF = 328;
2988 0 macro SWBSB_FLAGS = 0,0,8,0 %;
2989 0 macro SWBSV_ELLIPSIS = 0,0,1,0 %;
2990 0 macro SWBSV_BOUNDED = 0,1,1,0 %;
2991 0 macro SWBSV_WILD = 0,2,1,0 %;
2992 0 macro SWBSV_DELIMITER = 0,3,1,0 %;
2993 0 macro SWBSV_TRAVERSE = 0,4,1,0 %;
2994 0 macro SWBSV_FIRST = 0,5,1,0 %;
2995 0 macro SWBSV_ELLIPSIS_EXISTS = 0,6,1,0 %;
2996 0 macro SWBSV_VALID_DID = 0,7,1,0 %;
2997 0 macro SWBSB_PATLEN = 1,0,8,0 %;
2998 0 macro SWBSB_PPOS = 2,0,8,0 %;
2999 0 macro SWBSB_TOKENS_LEFT = 3,0,8,0 %;
3000 0 macro SWBSB_MINIMUM = 4,0,8,0 %;
3001 0 macro SWBSB_MAXIMUM = 5,0,8,0 %;
3002 0 macro SWBSB_FIRST_E = 6,0,8,0 %;
3003 0 macro SWBSB_DIRWCFLGS = 7,0,8,0 %;
3004 0 macro SWBSB_BID = 8,0,8,0 %;

```

: the DIRECTORYn descriptors must be contiguous

: ID

: flags (must be first)
: ellipsis
: ellipsis bounded
: wild name
: following delimiter
: should skip subtree
: first time through
: dir spec contains ...
: FIB DID is valid
: length of current token
: position in pattern
: number of non ... tokens left
: minimum level for success
: maximum level for success
: token ! of first ellipsis
: FWASB_DIRWCFLGS on entry
: block-ID

```

3005 0 macro SWB$B_BLN = 9,0,8,0 %;          ! length
3006 0 macro SWB$Q_PATTERN = 12,0,0,0 %;
3007 0 literal SWB$S_PATTERN = 8;            ! descriptor of pattern
3008 0 macro SWB$L_SCRATCH_PAT = 20,0,32,0 %; ! scratch copy of first longword
3009 0 macro SWB$T_SCRATCH_BUF = 24,0,0,0 %;
3010 0 literal SWB$S_SCRATCH_BUF = 48;        ! scratch temp buffer (same size as FW$T_WILD)
3011 0 macro SWB$T_PATTERN_BUF = 72,0,0,0 %;
3012 0 literal SWB$S_PATTERN_BUF = 256;      ! should be: FW$C_MAXDIRLEN-2,
3013 0

```

* Copyright (c) 1982, 1983
 * by DIGITAL Equipment Corporation, Maynard, Mass.

* This software is furnished under a license and may be used and copied
 * only in accordance with the terms of such license and with the
 * inclusion of the above copyright notice. This software or any other
 * copies thereof may not be provided or otherwise made available to any
 * other person. No title to and ownership of the software is hereby
 * transferred.

* The information in this software is subject to change without notice
 * and should not be construed as a commitment by DIGITAL Equipment
 * Corporation.

* DIGITAL assumes no responsibility for the use or reliability of its
 * software on equipment which is not supplied by DIGITAL.

 Created 15-SEP-1984 22:54:43 by VAX-11 SDL V2.0 Source: 15-SEP-1984 22:49:34 _\$255\$DUA28:[RMS.SRC]RMSSHR.

!*** MODULE \$FSBDEF ***

```

3040 0 literal SFSB$C_BID = 16;          ! sfsb code
3041 0 literal SFSB$C_FIX_LEN = 10;      ! 10 bytes of fixed size data
3042 0 literal SFSB$K_BLN = 68;          ! length of sfsb
3043 0 literal SFSB$C_BLN = 68;          ! length of sfsb
3044 0

```

keep the next two fields in same order as they are in FAB

```

3047 0 literal SFSB$S_SFSBDEF = 68;
3048 0 macro SFSB$Q_FILENAME = 0,0,0,0 %;
3049 0 literal SFSB$S_FILENAME = 8;      ! descriptor of shared file resource name.
3050 0 resource name is NODE, DEVICE, FILE_ID
3051 0 points to RESNAM, below
3052 0 macro SFSB$W_NAME_LEN = 0,0,16,0 %; ! subfield to address descriptor length field
3053 0 macro SFSB$L_ADDRESS = 4,0,32,0 %;  ! subfield to address descriptor address field
3054 0 macro SFSB$B_BID = 8,0,8,0 %;       ! block id
3055 0 macro SFSB$B_BLN = 9,0,8,0 %;       ! block length in longwords
3056 0 macro SFSB$B_CURMODE = 10,0,8,0 %;  ! Mode of the current lock
3057 0 macro SFSB$B_PREMODE = 11,0,8,0 %;  ! Mode of the previous lock
3058 0 macro SFSB$T_RESNAM = 12,0,0,0 %;
3059 0 literal SFSB$S_RESNAM = 32;          ! 32 bytes for name of shared resource
3060 0 macro SFSB$L_FAC_CODE = 12,0,32,0 %; ! RMS facility code (RMSS)
3061 0 macro SFSB$W_FID_NUM = 16,0,16,0 %; ! file id word one

```

```

3062 0 macro SFSB$W_FID_SEQ = 18,0,16,0 %;      ! file id word two
3063 0 macro SFSB$W_FID_RVN = 20,0,16,0 %;      ! file id word three
3064 0 macro SFSB$T_DEV_NAM = 22,0,0,0 %;
3065 0 literal SFSB$$ DEV_NAM = 22;              ! 22 bytes remain to hold device id (node$device_name)
3066 0 macro SFSB$L_LRSB = 44,0,32,0 %;          ! lock status block
3067 0 macro SFSB$W_STATUS = 44,0,16,0 %;        ! VMS status code
3068 0 macro SFSB$W_S_BITS = 46,0,16,0 %;        ! various status bits
3069 0 macro SFSB$L_LOCK_ID = 48,0,32,0 %;        ! second longword of LKSB is the lock id
3070 0 macro SFSB$L_LVB = 52,0,0,0 %;
3071 0 literal SFSB$$ LVB = 16;                  ! lock value block
3072 0 macro SFSB$B_FAC = 52,0,8,0 %;            ! fac bits from FAB
3073 0 macro SFSB$B_SHR = 53,0,8,0 %;            ! sharing bits (from FAB SHR field)
3074 0 macro SFSB$L_HBK = 60,0,32,0 %;          ! high block
3075 0 macro SFSB$L_EBK = 64,0,32,0 %;          ! end of file

```

*** MODULE \$GBSBDEF ***

GBSB field definitions - global buffer synchronization block

The GBSB contains the information necessary to determine if a global section is already open for a file on a given node, and is used for synchronizing access to the global section.

```

3085 0 gbsb:
3086 0      +-----+
3087 0      | FILE_NAME |
3088 0      +-----+
3089 0      | FLAGS | CURMODE | BLN | BID |
3090 0      +-----+
3091 0      | Resource Name |
3092 0      +-----+
3093 0      | Still to be def- | VMS status code |
3094 0      | ined status bits |
3095 0      +-----+
3096 0      | Lock Id. (Returned for new locks, |
3097 0      | input for conversions) |
3098 0      +-----+
3099 0      | GBC | GBREF |
3100 0      +-----+
3101 0      | GBS - size of GS in bytes |
3102 0      +-----+
3103 0      | spare |
3104 0      +-----+
3105 0      | spare |
3106 0      +-----+
3107 0      | spare |
3108 0      +-----+

```

```

3111 0 literal GBSB$C_BID = 9;                  ! gbsb code
3112 0 literal GBSB$M_NOTACCESSED = 1;
3113 0 literal GBSB$K_BLN = 68;                  ! length of gbsb
3114 0 literal GBSB$C_BLN = 68;                  ! length of gbsb
3115 0 literal GBSB$$GBSBDEF = 68;
3116 0 macro GBSB$Q_FILENAME = 0,0,0,0 %;
3117 0 literal GBSB$$FILENAME = 8;                ! descriptor of shared file resource name.
3118 0 ! resource name is NODE, DEVICE, FILE_ID

```

```

3119 0 ! points to RESNAM, below
3120 0 macro GBSB$W_NAME_LEN = 0,0,16,0 %; subfield to address descriptor length field
3121 0 macro GBSB$L_ADDRESS = 4,0,32,0 %; subfield to address descriptor address field
3122 0 macro GBSB$B_BID = 8,0,8,0 %; block id
3123 0 macro GBSB$B_BLN = 9,0,8,0 %; block length in longwords
3124 0 macro GBSB$B_CURMODE = 10,0,8,0 %; Mode of the current lock
3125 0 macro GBSB$B_FLAGS = 11,0,8,0 %; spare
3126 0 macro GBSB$V_NOTACCESSED = 11,0,1,0 %; Process has already decremented access count for GBS.
3127 0 macro GBSB$T_RESNAM = 12,0,0,0 %;
3128 0 literal GBSB$S_RESNAM = 32; 32 bytes for name of shared resource
3129 0 macro GBSB$L_LKSB = 44,0,32,0 %; lock status block
3130 0 macro GBSB$W_STATUS = 44,0,16,0 %; VMS status code
3131 0 macro GBSB$W_S_BITS = 46,0,16,0 %; various status bits
3132 0 macro GBSB$L_LOCK_ID = 48,0,32,0 %; second longword of LKSB is the lock id
3133 0 macro GBSB$W_GBC = 52,0,16,0 %; Number of global buffers in section.
3134 0 macro GBSB$W_GBREF = 54,0,16,0 %; Number of accessors to global section.
3135 0 macro GBSB$L_GS_SIZE = 56,0,32,0 %; Size of global section in bytes.
3136 0

```

```

*
* Copyright (c) 1982, 1983
* by DIGITAL Equipment Corporation, Maynard, Mass.
*
* This software is furnished under a license and may be used and copied
* only in accordance with the terms of such license and with the
* inclusion of the above copyright notice. This software or any other
* copies thereof may not be provided or otherwise made available to any
* other person. No title to and ownership of the software is hereby
* transferred.
*
* The information in this software is subject to change without notice
* and should not be construed as a commitment by DIGITAL Equipment
* Corporation.
*
* DIGITAL assumes no responsibility for the use or reliability of its
* software on equipment which is not supplied by DIGITAL.
*

```

```

*****
Created 15-SEP-1984 22:54:49 by VAX-11 SDL V2.0 Source: 15-SEP-1984 22:53:49 _$255$DUA28:[RMS.SRC]RMSDEF.
*****

```

*** MODULE \$RMSDEF ***

This SDL File Generated by VAX-11 Message V04-00 on 15-SEP-1984 22:53:50.83

.TITLE RMSDEF -RMC COMPLETION CODES

```

*
* COPYRIGHT (C) 1978, 1980, 1982, 1984 BY
* DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
* ALL RIGHTS RESERVED.
*
* THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
* ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
* INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER

```

```
* COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
* OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
* TRANSFERRED.
*
* THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
* AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
* CORPORATION.
*
* DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
* SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
*
*****
**
FACILITY: RMS

ABSTRACT:

THIS MODULE DEFINES ALL RMS COMPLETION CODES.

ENVIRONMENT:

THE MESSAGE TRANSLATOR MUST BE USED TO CONVERT RMSDEF.MSG INTO
RMSDEF.SDL. THE SDL TRANSLATOR MUST THEN BE USED TO CONVERT
RMSDEF.SDL INTO RMSDEF.MAR (AND RMSDEF.B32).

AUTHOR: LEO F. LAVERDURE, CREATION DATE: 10-DEC-1976

MODIFIED BY:

V03-024 RAS0314 RON SCHAEFER 21-JUN-1984
WORK OVER THE MESSAGES ONE MORE TIME: FIX TYPO IN OK_RNF,
DELETE ACPEOF ERROR, DELETE WSF ERROR.

V03-023 RAS0282 RON SCHAEFER 28-MAR-1984
MINOR TEXT CHANGES AND COMMENTS ADDED.
DELETE RMSS_COP ERROR.

V03-022 DAS0005 DAVID SOLOMON 19-MAR-1984
REMOVE RMSS_ECHO (NO LONGER NEEDED AS A RESULT OF NEW
IMPLEMENTATION FOR ECHO SYSS$INPUT -> SYSS$OUTPUT).

V03-021 JWT0150 JIM TEAGUE 01-FEB-1984
ADD RMSS$ IFF FOR TRYING TO WRITE-ACCESS A FILE ON
MAGTAPE THAT HAS NON-0 VALUE FOR ANSI BUFFER OFFSET.

V03-020 RAS0233 RON SCHAEFER 9-JAN-1984
ADD RMSS$ NOVALPRS ERROR FOR $SEARCH NOT PRECEDED BY
VALID $PARSE.

V03-019 JWT0148 JIM TEAGUE 20-DEC-1983
ADD A JNL XAB ERROR FOR CONFLICTING RU ATTRIBUTES
ON $CREATE: RMSS$_XCR.

V03-018 RAS0171 RON SCHAEFER 28-JUL-1983
ADD RMSS$ BUSY; A STRUCTURE-LESS ERROR (R0-ONLY STATUS)
REPLACING TEMP3.
```

N 13
15-Sep-1984 22:56:00
15-Sep-1984 22:55:58

VAX-11 Bliss-32 V4.0-742
_S255\$DUA28:[RMS.OBJ]RMS.R32;1

Page 62
(9)

V03-017 DAS0004 DAVID SOLOMON 28-JUN-1983
ADD RMSS_FILEPURGED; ADD RMSS_ACPEOF FOR ZALEWSKI.

V03-016 KPL0007 PETER LIEBERWIRTH 8-JUN-1983
FIX SPELLING IN JNL ERROR MESSAGES, ADD CJF ERROR WHICH
WILL SOON SUBSUME COP AND CQE.

V03-015 DAS0003 DAVID SOLOMON 18-FEB-1983
ADD XNF (REPLACING TEMP9), TMR (NEW), LWC (NEW).

V03-014 KBT0497 KEITH B. THOMPSON 18-FEB-1983
ADD INCOMPSTR ERROR (REUSE OF TEMP1 SLOT)

V03-013 RAS0120 RON SCHAEFER 2-FEB-1983
ADD ECHO PSEUDO-STATUS TO SUPPORT ECHO OF SYSSINPUT
TO SYSSOUTPUT.

V03-012 JWH0174 JEFFRY W. HORN 24-JAN-1983
ADD CQE AND COP ERRORS.

V03-011 KPL0006 PETER LIEBERWIRTH 7-JAN-1983
ADD OK_RULK FOR RECOVERY UNIT SUPPORT.
ADD REENT ERROR FOR \$RENAME.

V03-009 JWH0153 JEFFREY W. HORN 8-DEC-1982
ADD NRU ERROR.

V03-008 JWH0152 JEFFREY W. HORN 8-DEC-1982
ADD JNS ERROR.

V03-007 MCN0002 MARIA DEL C. NASR 15-NOV-1982
REPLACE ORD ERROR CODE ELIMINATED BY MCN0001, SINCE
THE NETWORK CODE REFERENCES IT.

V03-006 MCN0001 MARIA DEL C. NASR 26-OCT-1982
PRINT KEY OF REFERENCE OR AREA IDENTIFICATION NUMBER
INSTEAD OF XAB ADDRESS FOR ERRORS RETURNED DURING
INDEXED FILE CREATION. ALSO RENAME ALL UNUSED ERROR
CODES TO TEMP.

V03-005 JWH0102 JEFFREY W. HORN 24-SEP-1982
ADD RUP ERROR.

V03-004 RAS0095 RON SCHAEFER 7-SEP-1982
ADD OVRDSKQUOTA ERROR.

V03-002 JWH0002 JEFFREY W. HORN 26-JUL-1982
CONVERT TO .MSG FORMAT. ADD RUM ERROR.

V03-001 JWH0001 JEFFREY W. HORN 20-JUL-1982
ADD JNF, JOP, AND NOJ ERRORS.

V02-042 KPL0005 PETER LIEBERWIRTH 4-FEB-1982
ADD ERROR MESSAGES RMSS_EXENQLM AND RMSS_DEADLOCK
CORRECTLY.

B 14
15-Sep-1984 22:56:00
15-Sep-1984 22:55:58

VAX-11 Bliss-32 V4.0-742
_S255\$DUA28:[RMS.OBJ]RMS.R32;1

Page 63
(9)

3290 0
3291 0
3292 0
3293 0
3294 0
3295 0
3296 0
3297 0
3298 0
3299 0
3300 0
3301 0
3302 0
3303 0
3304 0
3305 0
3306 0
3307 0
3308 0
3309 0
3310 0
3311 0
3312 0
3313 0
3314 0
3315 0
3316 0
3317 0
3318 0
3319 0
3320 0
3321 0
3322 0
3323 0
3324 0
3325 0
3326 0
3327 0
3328 0
3329 0
3330 0
3331 0
3332 0
3333 0
3334 0
3335 0
3336 0
3337 0
3338 0
3339 0
3340 0
3341 0
3342 0
3343 0
3344 0
3345 0
3346 0

V02-041 CDS0001 C D SAETHER 24-JAN-1982
ADD GBC AND CRMP ERRORS.

V02-040 JAK0069 J A KRYCKA 15-JAN-1982
ADD SUPPORT AND NETFAIL ERROR CODES.

V02-039 JAK0068 J A KRYCKA 31-DEC-1981
MODIFY TEXT FOR SUP, NET, BUG_DAP, AND ENV ERROR MESSAGES.

V02-038 LJA0001 LAURIE ANDERSON 20-DEC-1981
MODIFIED SOME MESSAGES TO READ BETTER.

V02-037 JAK0063 J A KRYCKA 31-AUG-1981
ADD CRE_STM SUCCESS CODE.

V02-036 KPL0004 PETER LIEBERWIRTH 13-JAN-1981
ADD ENQ SEVERE ERROR MESSAGE, TO INDICATE \$ENQ/\$DEQ FAILURE.
ALSO ADD SUCCESS CODES OK_RRL, AND OK_WAT.

V02-035 REFORMAT J A KRYCKA 30-JUL-1980

--
literal RMSS\$ FACILITY = 1;
literal RMSS\$ STVSTATUS = 14; ! MOVE TO BIT 14 OF THE
STATUS CODE IT INDICATES
THAT STV CONTAINS A SECONDARY
STATUS CODE.

literal RMSS\$ SUC = 65537;
literal RMSS\$ NORMAL = 65537;

SUCCESS CODES

-
BIT 16 = BIT 15 = 1
literal RMSS\$ STALL = 98305;
(NOTE: USER NEVER RECEIVES THIS CODE)
literal RMSS\$ PENDING = 98313;
literal RMSS\$ OK_DUP = 98321;
literal RMSS\$ OK_IDX = 98329;
(RECORD HAS BEEN INSERTED, BUT INDEX STRUCTURE IS NO LONGER
OPTIMAL.)
literal RMSS\$ OK_RLK = 98337;
(BECAUSE LOCKER SET RLK IN ROP FIELD WHEN RECORD WAS LOCKED.)
literal RMSS\$ OK_RRL = 98345;
(BECAUSE READER SET RRL IN ROP FIELD WHEN ACCESSING RECORD.)
(THIS CODE USED TO BE OK_RRV)
OK_RRV ;MSG <record was inserted successfully in primary>
(BUT IT MAY NOT BE ACCESSIBLE VIA ONE-OR-MORE SECONDARY KEYS,
AND NOT BY RFA ADDRESSING. FILE RE-ORGANIZATION RECOMMENDED!)
literal RMSS\$ KFF = 98353;
literal RMSS\$ OK_ALK = 98361;
literal RMSS\$ OK_DEL = 98369;
literal RMSS\$ OK_RNF = 98377;
literal RMSS\$ OK_LIM = 98385;
literal RMSS\$ OK_NOP = 98393;

C 14
15-Sep-1984 22:56:00
15-Sep-1984 22:55:58

VAX-11 Bliss-32 V4.0-742
_S255\$DUA28:[RMS.OBJ]RMS.R32;1

Page 64
(9)

```
3347 0 Literal RMS$ OK_WAT = 98401;
3348 0 (BECAUSE WAT-BIT IN ROP WAS SET AND RECORD WAS ALREADY
3349 0 LOCKED.)
3350 0 Literal RMS$ CRE_STM = 98409;
3351 0 Literal RMS$ OK_RULK = 98417;
3352 0 +
3353 0
3354 0 SUCCESS CODES PASSED THRU FROM DRIVERS AND ACP - BIT 15 = 0
3355 0
3356 0 -
3357 0 Literal RMS$ CONTROLC = 67153; ! TERMINAL I/O ABORTED DUE TO CTRL/C
3358 0 Literal RMS$ CONTROL0 = 67081;
3359 0 TERMINAL I/O ABORTED DUE TO CTRL/O
3360 0 Literal RMS$ CONTROLY = 67089;
3361 0 TERMINAL I/O ABORTED DUE TO CTRL/Y
3362 0 Literal RMS$ CREATED = 67097;
3363 0 FILE WAS CREATED, NOT OPENED
3364 0 Literal RMS$ SUPERSEDE = 67121;
3365 0 CREATED FILE SUPERSEDED EXISTING VERSION
3366 0 Literal RMS$ OVRDSKQUOTA = 67177;
3367 0 DISK USAGE EXCEEDS DISK QUOTA
3368 0 Literal RMS$ FILEPURGED = 67193;
3369 0 CREATE FICE CAUSED OLDEST FILE
3370 0 VERSION TO BE PURGED
3371 0 +
3372 0
3373 0 WARNING CODES
3374 0
3375 0 -
3376 0 BIT 16 = BIT 15 = 1, BIT 14 = 0
3377 0 Literal RMS$ BOF = 98712;
3378 0 Literal RMS$ RNL = 98720;
3379 0 Literal RMS$ RTB = 98728;
3380 0 Literal RMS$ TMO = 98736;
3381 0 Literal RMS$ TNS = 98744;
3382 0 Literal RMS$ BES = 98752;
3383 0 Literal RMS$ PES = 98760;
3384 0 +
3385 0
3386 0 ERROR CODES - WITHOUT STV
3387 0
3388 0 -
3389 0 BIT 16 = BIT 15 = 1, BIT 14 = 0
3390 0 Literal RMS$ ACT = 98906;
3391 0 Literal RMS$ DEL = 98914;
3392 0 Literal RMS$ INCOMPSHR = 98922;
3393 0 Literal RMS$ DNR = 98930;
3394 0 Literal RMS$ EOF = 98938;
3395 0 Literal RMS$ FEX = 98946;
3396 0 Literal RMS$ FLK = 98954;
3397 0 Literal RMS$ FNF = 98962;
3398 0 Literal RMS$ PRV = 98970;
3399 0 Literal RMS$ REX = 98978;
3400 0 Literal RMS$ RLK = 98986;
3401 0 Literal RMS$ RNF = 98994;
3402 0 (RECORD NEVER WAS IN FILE, OR HAS BEEN DELETED.)
3403 0 Literal RMS$ WLK = 99002;
```

```
3404 0 literal RMSS_EXP = 99010;
3405 0 literal RMSS_NMF = 99018;
3406 0 literal RMSS_SUP = 99026;
3407 0 (NOTE THAT SUPPORT HAS REPLACED SUP EXCEPT WHEN RMS CANNOT
3408 0 MAP THE DAP STATUS CODE INTO A FAL STATUS CODE.)
3409 0 (NOTE ALSO THAT SUP SHOULD HAVE BEEN DEFINED AS A SEVERE
3410 0 ERROR CODE. HOWEVER, SUPPORT IS A SEVERE ERROR CODE.)
3411 0 literal RMSS_RSA = 99034;
3412 0 literal RMSS_CRC = 99042;
3413 0 literal RMSS_WCC = 99050;
3414 0 literal RMSS_IDR = 99058;
3415 0 literal RMSS_LWC = 99066;
3416 0 literal RMSS_XCR = 99074;
3417 0 literal RMSS_NOVALPRS = 99082;
3418 0 +
3419 0
3420 0 ERROR CODES - WITH STV ERROR CODE
3421 0
3422 0 -
3423 0 BIT 16 = BIT 15 = BIT 14 = 1
3424 0 literal RMSS_ACC = 114690;
3425 0 literal RMSS_CRE = 114698;
3426 0 literal RMSS_DAC = 114706;
3427 0 literal RMSS_ENT = 114714;
3428 0 literal RMSS_EXT = 114722;
3429 0 literal RMSS_FND = 114730;
3430 0 literal RMSS_MKD = 114738;
3431 0 literal RMSS_DPE = 114746;
3432 0 literal RMSS_SPL = 114754;
3433 0 literal RMSS_DNF = 114762;
3434 0 literal RMSS_JNF = 114770;
3435 0 +
3436 0
3437 0 SEVERE ERROR CODES - WITHOUT STV
3438 0
3439 0 -
3440 0 BIT 16 = BIT 15 = 1, BIT 14 = 0
3441 0 literal RMSS_TEMPO = 99308;
3442 0 literal RMSS_AID = 99316;
3443 0 literal RMSS_ALN = 99324;
3444 0 literal RMSS_ALQ = 99332;
3445 0 literal RMSS_ANI = 99340;
3446 0 literal RMSS_AOP = 99348;
3447 0 literal RMSS_BKS = 99356;
3448 0 literal RMSS_BKZ = 99364;
3449 0 literal RMSS_BLN = 99372;
3450 0 literal RMSS_BUG = 99380;
3451 0 literal RMSS_BUG_DDI = 99388;
3452 0 literal RMSS_BUG_DAP = 99396;
3453 0 literal RMSS_BUG_XX1 = 99404;
3454 0 literal RMSS_BUG_XX2 = 99412;
3455 0 literal RMSS_BUG_XX3 = 99420;
3456 0 literal RMSS_BUG_XX4 = 99428;
3457 0 literal RMSS_BUG_XX5 = 99436;
3458 0 literal RMSS_BUG_XX6 = 99444;
3459 0 literal RMSS_BUG_XX7 = 99452;
3460 0 literal RMSS_BUG_XX8 = 99460;
```

E 14
15-Sep-1984 22:56:00
15-Sep-1984 22:55:58

VAX-11 Bliss-32 V4.0-742
_S255\$DUA28:[RMS.OBJ]RMS.R32;1

Page 66
(9)

```
3461 0 literal RMSS-BUSY = 99468;
3462 0 literal RMSS-CCR = 99476;
3463 0 literal RMSS-CHG = 99484;
3464 0 literal RMSS-CHK = 99492;
3465 0 literal RMSS-COD = 99500;
3466 0 literal RMSS-CUR = 99508;
3467 0 literal RMSS-DAN = 99516;
3468 0 literal RMSS-DEV = 99524;
3469 0 literal RMSS-DIR = 99532;
3470 0 literal RMSS-DME = 99540;
3471 0 literal RMSS-DNA = 99548;
3472 0 literal RMSS-DTP = 99556;
3473 0 literal RMSS-DUP = 99564;
3474 0 literal RMSS-DVI = 99572;
3475 0 literal RMSS-ESA = 99580;
3476 0 literal RMSS-ESS = 99588;
3477 0 literal RMSS-FAB = 99596;
3478 0 literal RMSS-FAC = 99604;
3479 0 literal RMSS-FLG = 99612;
3480 0 literal RMSS-FNA = 99620;
3481 0 literal RMSS-FNM = 99628;
3482 0 literal RMSS-FSZ = 99636;
3483 0 literal RMSS-FOP = 99644;
3484 0 literal RMSS-FUL = 99652;
3485 0 literal RMSS-IAL = 99660;
3486 0 literal RMSS-IAN = 99668;
3487 0 literal RMSS-IDX = 99676;
3488 0 literal RMSS-IFI = 99684;
3489 0 literal RMSS-IMX = 99692;
3490 0 literal RMSS-IOP = 99700;
3491 0 literal RMSS-IRC = 99708;
3492 0 literal RMSS-ISI = 99716;
3493 0 literal RMSS-KBF = 99724;
3494 0 literal RMSS-KEY = 99732;
3495 0 literal RMSS-KRF = 99740;
3496 0 literal RMSS-KSZ = 99748;
3497 0 literal RMSS-LAN = 99756;
3498 0 literal RMSS-TEMP1 = 99764;
3499 0 literal RMSS-LNE = 99772;
3500 0 literal RMSS-TEMP2 = 99780;
3501 0 literal RMSS-MRN = 99788;
3502 0 literal RMSS-MRS = 99796;
3503 0 literal RMSS-NAM = 99804;
3504 0 literal RMSS-NEF = 99812;
3505 0 literal RMSS-TEMP3 = 99820;
3506 0 literal RMSS-NOD = 99828;
3507 0 literal RMSS-NPK = 99836;
3508 0 literal RMSS-ORD = 99844;
3509 0 literal RMSS-ORG = 99852;
3510 0 literal RMSS-PBF = 99860;
3511 0 literal RMSS-PLG = 99868;
3512 0 literal RMSS-POS = 99876;
3513 0 literal RMSS-TEMP4 = 99884;
3514 0 literal RMSS-QUO = 99892;
3515 0 literal RMSS-RAB = 99900;
3516 0 literal RMSS-RAC = 99908;
3517 0 literal RMSS-RAT = 99916;
```

! ! NOT USED AS OF V4

```
3518 0 literal RMSS_RBF = 99924;
3519 0 literal RMSS_RFA = 99932;
3520 0 literal RMSS_RFM = 99940;
3521 0 literal RMSS_RHB = 99948;
3522 0 literal RMSS_RLF = 99956;
3523 0 literal RMSS_ROP = 99964;
3524 0 literal RMSS_RRV = 99972;
3525 0 literal RMSS_RVU = 99980;
3526 0 literal RMSS_RSS = 99988;
3527 0 literal RMSS_RST = 99996;
3528 0 literal RMSS_RSZ = 100004;
3529 0 literal RMSS_SEQ = 100012;
3530 0 literal RMSS_SHR = 100020;
3531 0 literal RMSS_SIZ = 100028;
3532 0 literal RMSS_SQO = 100036;
3533 0 literal RMSS_TEMP5 = 100044;
3534 0 literal RMSS_SYN = 100052;
3535 0 literal RMSS_TRE = 100060;
3536 0 literal RMSS_TYP = 100068;
3537 0 literal RMSS_UBF = 100076;
3538 0 literal RMSS_USZ = 100084;
3539 0 literal RMSS_VER = 100092;
3540 0 literal RMSS_XNF = 100100;
3541 0 literal RMSS_XAB = 100108;
3542 0 literal RMSS_ESL = 100116;
3543 0 literal RMSS_TEMP6 = 100124;
3544 0 ! (THIS CODE USED TO BE WSF)
3545 0 ! WSF <working set full (cannot lock buffers in working set)>
3546 0 literal RMSS_ENV = 100132;
3547 0 literal RMSS_PLV = 100140;
3548 0 literal RMSS_MBC = 100148;
3549 0 literal RMSS_RSL = 100156;
3550 0 literal RMSS_WLD = 100164;
3551 0 literal RMSS_NET = 100172;
3552 0 ! (NOTE THAT NETFAIL HAS REPLACED NET EXCEPT WHEN RMS CANNOT
3553 0 ! MAP THE DAP STATUS CODE INTO A FAL STATUS CODE.)
3554 0 literal RMSS_IBF = 100180;
3555 0 literal RMSS_REF = 100188;
3556 0 literal RMSS_IFL = 100196;
3557 0 literal RMSS_DFL = 100204;
3558 0 literal RMSS_KNM = 100212;
3559 0 literal RMSS_IBK = 100220;
3560 0 literal RMSS_KSI = 100228;
3561 0 literal RMSS_LEX = 100236;
3562 0 literal RMSS_SEG = 100244;
3563 0 literal RMSS_SNE = 100252;
3564 0 literal RMSS_SPE = 100260;
3565 0 literal RMSS_UPI = 100268;
3566 0 literal RMSS_ACS = 100276;
3567 0 literal RMSS_STR = 100284;
3568 0 literal RMSS_FTM = 100292;
3569 0 literal RMSS_GBC = 100300;
3570 0 literal RMSS_DEADLOCK = 100308;
3571 0 literal RMSS_EXENQLM = 100316;
3572 0 literal RMSS_JOP = 100324;
3573 0 literal RMSS_RUM = 100332;
3574 0 literal RMSS_JNS = 100340;
```

!! NOT USED AS OF V2

! ! NOT USED AS OF V4
! ! NOT USED AS OF V4

```

3575 0 literal RMSS_NRU = 100348;
3576 0 literal RMSS_IFF = 100356;          ! +
3577 0
3578 0 SEVERE ERRORS - WITH STV ERROR CODE
3579 0
3580 0 -
3581 0 BIT 16 = BIT 15 = BIT 14 = 1
3582 0 literal RMSS_ATR = 114892;
3583 0 literal RMSS_ATW = 114900;
3584 0 literal RMSS_CCF = 114908;
3585 0 literal RMSS_CDA = 114916;
3586 0 literal RMSS_CHN = 114924;
3587 0 literal RMSS_RER = 114932;
3588 0 literal RMSS_RMV = 114940;
3589 0 literal RMSS_RPL = 114948;
3590 0 literal RMSS_SYS = 114956;
3591 0 literal RMSS_WER = 114964;
3592 0 literal RMSS_WPL = 114972;
3593 0 literal RMSS_IFA = 114980;
3594 0 literal RMSS_WBE = 114988;
3595 0 literal RMSS_ENQ = 114996;
3596 0 literal RMSS_NETFAIL = 115004;
3597 0 literal RMSS_SUPPORT = 115012;
3598 0 literal RMSS_CRMP = 115020;
3599 0 literal RMSS_NOJ = 115028;
3600 0 literal RMSS_REENT = 115036;
3601 0 literal RMSS_CIF = 115044;
3602 0 literal RMSS_TMR = 115052;
3603 0
3604 0 *****
3605 0 *
3606 0 * Copyright (c) 1982, 1983
3607 0 * by DIGITAL Equipment Corporation, Maynard, Mass.
3608 0 *
3609 0 * This software is furnished under a license and may be used and copied
3610 0 * only in accordance with the terms of such license and with the
3611 0 * inclusion of the above copyright notice. This software or any other
3612 0 * copies thereof may not be provided or otherwise made available to any
3613 0 * other person. No title to and ownership of the software is hereby
3614 0 * transferred.
3615 0 *
3616 0 * The information in this software is subject to change without notice
3617 0 * and should not be construed as a commitment by DIGITAL Equipment
3618 0 * Corporation.
3619 0 *
3620 0 * DIGITAL assumes no responsibility for the use or reliability of its
3621 0 * software on equipment which is not supplied by DIGITAL.
3622 0 *
3623 0 *****
3624 0 *****
3625 0 (created 15-SEP-1984 22:55:02 by VAX-11 SDL V2.0 Source: 15-SEP-1984 22:49:39 _$255$DUA28:[RMS.SRC]RMSUSR.
3626 0 *****
3627 0
3628 0
3629 0 *** MODULE $FABDEF ***
3630 0 *****
3631 0 the fields thru ctx must not be modified due to

```

```
3632 0 ! commonality between fab/rab/xab
3633 0 literal FABSC_BID = 3; . code for fab
3634 0 literal FABSM_PPF_RAT = 16320;
3635 0 literal FABSM_PPF_IND = 16384;
3636 0 literal FABSM_MXV = 2;
3637 0 literal FABSM_SUP = 4;
3638 0 literal FABSM_TMP = 8;
3639 0 literal FABSM_TMD = 16;
3640 0 literal FABSM_DFW = 32;
3641 0 literal FABSM_SQO = 64;
3642 0 literal FABSM_RWO = 128;
3643 0 literal FABSM_POS = 256;
3644 0 literal FABSM_WCK = 512;
3645 0 literal FABSM_NEF = 1024;
3646 0 literal FABSM_RWC = 2048;
3647 0 literal FABSM_DMO = 4096;
3648 0 literal FABSM_SPL = 8192;
3649 0 literal FABSM_SCF = 16384;
3650 0 literal FABSM_DLT = 32768;
3651 0 literal FABSM_NFS = 65536;
3652 0 literal FABSM_UFO = 131072;
3653 0 literal FABSM_PPF = 262144;
3654 0 literal FABSM_INP = 524288;
3655 0 literal FABSM_CTG = 1048576;
3656 0 literal FABSM_CBT = 2097152;
3657 0 literal FABSM_RCK = 8388608;
3658 0 literal FABSM_NAM = 16777216;
3659 0 literal FABSM_CIF = 33554432;
3660 0 literal FABSM_ESC = 134217728;
3661 0 literal FABSM_TEF = 268435456;
3662 0 literal FABSM_OFP = 536870912;
3663 0 literal FABSM_KFO = 1073741824;
3664 0 literal FABSM_PUT = 1;
3665 0 literal FABSM_GET = 2;
3666 0 literal FABSM_DEL = 4;
3667 0 literal FABSM_UPD = 8;
3668 0 literal FABSM_TRN = 16;
3669 0 literal FABSM_BIO = 32;
3670 0 literal FABSM_BRO = 64;
3671 0 literal FABSM_EXE = 128;
3672 0 literal FABSM_SHRPUT = 1;
3673 0 literal FABSM_SHRGET = 2;
3674 0 literal FABSM_SHRDEL = 4;
3675 0 literal FABSM_SHRUPD = 8;
3676 0 literal FABSM_MSIF = 16;
3677 0 literal FABSM_NIL = 32;
3678 0 literal FABSM_UPI = 64;
3679 0 literal FABSC_SEQ = 0;
3680 0 literal FABSC_REL = 16;
3681 0 literal FABSC_IDX = 32;
3682 0 literal FABSC_HSH = 48;
3683 0 literal FABSM_FTN = 1;
3684 0 literal FABSM_CR = 2;
3685 0 literal FABSM_PRN = 4;
3686 0 literal FABSM_BLK = 8;
3687 0 literal FABSC_RFM_DFLT = 2;
3688 0 literal FABSC_UDF = 0;

! sequential
! relative
! indexed
! hashed

! var len is default
! undefined (also stream binary)
```

1 14
15-Sep-1984 22:56:00
15-Sep-1984 22:55:58

VAX-11 Bliss-32 V4.0-742
_S255SDUA28:[RMS.OBJ]RMS.R32;1

Page 70
(9)

3689	0	literal FABSC_FIX = 1;	fixed length records
3690	0	literal FABSC_VAR = 2;	variable length records
3691	0	literal FABSC_VFC = 3;	variable fixed control
3692	0	literal FABSC_STM = 4;	RMS-11 stream (valid only for sequential org)
3693	0	literal FABSC_STMLF = 5;	LF stream (valid only for sequential org)
3694	0	literal FABSC_STMCR = 6;	CR stream (valid only for sequential org)
3695	0	literal FABSC_MAXRFM = 6;	maximum rfm supported
3696	0	literal FABSM_RU = 1;	
3697	0	literal FABSM_AI = 2;	
3698	0	literal FABSM_BI = 4;	
3699	0	literal FABSK_BLN = 80;	length of fab
3700	0	literal FABSC_BLN = 80;	length of fab
3701	0	literal FABSS_FABDEF = 80;	
3702	0	macro FABSB_BID = 0,0,8,0 %;	block id
3703	0	macro FABSB_BLN = 1,0,8,0 %;	block len
3704	0	macro FABSR_IFI_OVERLAY = 2,0,16,0 %;	
3705	0	macro FABSW_IFI = 2,0,16,0 %;	internal file index
3706	0	macro FABSR_IFI_BITS = 2,0,16,0 %;	
3707	0	macro FABSV_PPF_RAT = 2,6,8,0 %;	
3708	0	literal FABSS_PPF_RAT = 8;	rat value for process-permanent files
3709	0	macro FABSV_PPF_IND = 2,14,1,0 %;	indirect access to process-permanent file
3710	0	! (i.e., restricted operations)	
3711	0	macro FABSR_FOP_OVERLAY = 4,0,32,0 %;	
3712	0	macro FABSL_FOP = 4,0,32,0 %;	file options
3713	0	macro FABSR_FOP_BITS = 4,0,32,0 %;	
3714	0	macro FABSV_MXV = 4,1,1,0 %;	maximize version number
3715	0	macro FABSV_SUP = 4,2,1,0 %;	supersede existing file
3716	0	macro FABSV_TMP = 4,3,1,0 %;	create temporary file
3717	0	macro FABSV_TMD = 4,4,1,0 %;	create temp file marked for delete
3718	0	macro FABSV_DFW = 4,5,1,0 %;	deferred write (rel and idx)
3719	0	macro FABSV_SQO = 4,6,1,0 %;	sequential access only
3720	0	macro FABSV_RWO = 4,7,1,0 %;	rewind mt on open
3721	0	macro FABSV_POS = 4,8,1,0 %;	use next magtape position
3722	0	macro FABSV_WCK = 4,9,1,0 %;	write checking
3723	0	macro FABSV_NEF = 4,10,1,0 %;	inhibit end of file positioning
3724	0	macro FABSV_RWC = 4,11,1,0 %;	rewind mt on close
3725	0	macro FABSV_DMO = 4,12,1,0 %;	dismount mt on close (not implemented)
3726	0	macro FABSV_SPL = 4,13,1,0 %;	spool file on close
3727	0	macro FABSV_SCF = 4,14,1,0 %;	submit command file on close
3728	0	macro FABSV_DLT = 4,15,1,0 %;	delete sub-option
3729	0	macro FABSV_NFS = 4,16,1,0 %;	non-file structured operation
3730	0	macro FABSV_UFO = 4,17,1,0 %;	user file open - no rms operations
3731	0	macro FABSV_PPF = 4,18,1,0 %;	process permanent file (pio segment)
3732	0	macro FABSV_INP = 4,19,1,0 %;	process-permanent file is 'input'
3733	0	macro FABSV_CTG = 4,20,1,0 %;	contiguous extension
3734	0	macro FABSV_CBT = 4,21,1,0 %;	contiguous best try
3735	0	macro FABSV_RCK = 4,23,1,0 %;	read checking
3736	0	macro FABSV_NAM = 4,24,1,0 %;	use name block dvi, did, and/or fid fields for open
3737	0	macro FABSV_CIF = 4,25,1,0 %;	create if non-existent
3738	0	macro FABSV_ESC = 4,27,1,0 %;	'escape' to non-standard function (\$modify)
3739	0	macro FABSV_TEF = 4,28,1,0 %;	truncate at eof on close (write-accessed seq. disk file only)
3740	0	macro FABSV_OFP = 4,29,1,0 %;	output file parse (only name type sticky)
3741	0	macro FABSV_KFO = 4,30,1,0 %;	known file open (image activator only release 1)
3742	0	macro FABSL_STS = 8,0,32,0 %;	status
3743	0	macro FABSL_STV = 12,0,32,0 %;	status value
3744	0	macro FABSL_ALQ = 16,0,32,0 %;	allocation quantity
3745	0	macro FABSW_DEQ = 20,0,16,0 %;	default allocation quantity


```

3746 0 macro FABSR_FAC_OVERLAY = 22,0,8,0 %;
3747 0 macro FABSB_FAC = 22,0,8,0 %;
3748 0 macro FABSR_FAC_BITS = 22,0,8,0 %;
3749 0 macro FABSV_PUT = 22,0,1,0 %;
3750 0 macro FABSV_GET = 22,1,1,0 %;
3751 0 macro FABSV_DEL = 22,2,1,0 %;
3752 0 macro FABSV_UPD = 22,3,1,0 %;
3753 0 macro FABSV_TRN = 22,4,1,0 %;
3754 0 macro FABSV_BIO = 22,5,1,0 %;
3755 0 macro FABSV_BRO = 22,6,1,0 %;
3756 0 macro FABSV_EXE = 22,7,1,0 %;
3757 0 ! ufo must also be set)
3758 0 macro FABSR_SHR_OVERLAY = 23,0,8,0 %;
3759 0 macro FABSB_SHR = 23,0,8,0 %;
3760 0 macro FABSR_SHR_BITS = 23,0,8,0 %;
3761 0 macro FABSV_SHRPUT = 23,0,1,0 %;
3762 0 macro FABSV_SHRGET = 23,1,1,0 %;
3763 0 macro FABSV_SHRDEL = 23,2,1,0 %;
3764 0 macro FABSV_SHRUPD = 23,3,1,0 %;
3765 0 macro FABSV_MSE = 23,4,1,0 %;
3766 0 macro FABSV_NIL = 23,5,1,0 %;
3767 0 macro FABSV_UPI = 23,6,1,0 %;
3768 0 ! writers to seq. files)
3769 0 macro FABSL_CTX = 24,0,32,0 %;
3770 0 ! -----
3771 0 macro FABSB_RTV = 28,0,8,1 %;
3772 0 macro FABSR_ORG_OVERLAY = 29,0,8,0 %;
3773 0 macro FABSB_ORG = 29,0,8,0 %;
3774 0 macro FABSR_ORG_BITS = 29,0,8,0 %;
3775 0 macro FABSV_ORG = 29,4,4,0 %;
3776 0 literal FABSS_ORG = 4;
3777 0 macro FABSR_RAT_OVERLAY = 30,0,8,0 %;
3778 0 macro FABSB_RAT = 30,0,8,0 %;
3779 0 macro FABSR_RAT_BITS = 30,0,8,0 %;
3780 0 macro FABSV_FTN = 30,0,1,0 %;
3781 0 macro FABSV_CR = 30,1,1,0 %;
3782 0 macro FABSV_PRN = 30,2,1,0 %;
3783 0 macro FABSV_BLK = 30,3,1,0 %;
3784 0 macro FABSB_RFM = 31,0,8,0 %;
3785 0 macro FABSL_JNL = 32,0,32,0 %;
3786 0 macro FABSL_XAB = 36,0,32,0 %;
3787 0 macro FABSL_NAM = 40,0,32,0 %;
3788 0 macro FABSL_FNA = 44,0,32,0 %;
3789 0 macro FABSL_DNA = 48,0,32,0 %;
3790 0 macro FABSB_FNS = 52,0,8,0 %;
3791 0 macro FABSB_DNS = 53,0,8,0 %;
3792 0 macro FABSW_MRS = 54,0,16,0 %;
3793 0 macro FABSL_MRN = 56,0,32,0 %;
3794 0 macro FABSW_BLS = 60,0,16,0 %;
3795 0 macro FABSB_BKS = 62,0,8,0 %;
3796 0 macro FABSB_FSZ = 63,0,8,0 %;
3797 0 macro FABSL_DEV = 64,0,32,0 %;
3798 0 macro FABSL_SDC = 68,0,32,0 %;
3799 0 macro FABSW_GBC = 72,0,16,0 %;
3800 0 macro FABSR_ACMODES_OVERLAY = 74,0,8,0 %;
3801 0 macro FABSB_ACMODES = 74,0,8,0 %;
3802 0 macro FABSR_ACMODES_BITS = 74,0,8,0 %;

```

file access

put access
get access
delete access
update access
truncate access
block i/o access
block and record i/o access
execute access (caller must be exec or kernel mode,

file sharing

put access
get access
delete access
update access
multi-stream connects enabled
no sharing
user provided interlocking (allows multiple

user context

retrieval window size

file organization

record format

fortran carriage-ctl
lf-record-cr carriage ctl
print-file carriage ctl
records don't cross block boundaries
record format
lcb address
xab address
nam block address
file name string address
default file name string addr
file name string size
default name string size
maximum record size
maximum record number
blocksize for tape
bucket size
fixed header size
device characteristics
spooling device characteristics
Global buffer count

agent access modes

```

3803 0 macro FABSV_LNM_MODE = 74,0,2,0 %;
3804 0 literal FABSS_LNM_MODE = 2; ! ACMODE for log nams
3805 0 macro FABSV_CHAN_MODE = 74,2,2,0 %;
3806 0 literal FABSS_CHAN_MODE = 2; ! ACMODE for channel
3807 0 macro FABSV_FILE_MODE = 74,4,2,0 %;
3808 0 literal FABSS_FILE_MODE = 2; ! ACMODE to use for determining file accessibility
3809 0 macro FABSR_RCF_OVERLAY = 75,0,8,0 %; ! recovery control flags
3810 0 macro FABSB_RCF = 75,0,8,0 %;
3811 0 macro FABSR_RCF_BITS = 75,0,8,0 %;
3812 0 macro FABSV_RU = 75,0,1,0 %; ! recovery unit recovery
3813 0 macro FABSV_AI = 75,1,1,0 %; ! after image recovery
3814 0 macro FABSV_BI = 75,2,1,0 %; ! before image recovery
3815 0
3816 0 *** MODULE $RABDEF ***
3817 0
3818 0 record access block (rab) definitions
3819 0
3820 0 there is one rab per connected stream
3821 0 it is used for all communications between the user
3822 0 and rms concerning operations on the stream
3823 0
3824 0 ++++++
3825 0 the fields thru ctx cannot be changed due to commonality
3826 0 with the fab
3827 0
3828 0 literal RABSC_BID = 1; ! code for rab
3829 0 literal RABSM_PPF_RAT = 16320;
3830 0 literal RABSM_PPF_IND = 16384;
3831 0 literal RABSM_ASY = 1;
3832 0 literal RABSM_TPT = 2;
3833 0 literal RABSM_REA = 4;
3834 0 literal RABSM_RRL = 8;
3835 0 literal RABSM_UIF = 16;
3836 0 literal RABSM_MAS = 32;
3837 0 literal RABSM_FDL = 64;
3838 0 literal RABSM_HSH = 128;
3839 0 literal RABSM_EOF = 256;
3840 0 literal RABSM_RAH = 512;
3841 0 literal RABSM_WBH = 1024;
3842 0 literal RABSM_BIO = 2048;
3843 0 literal RABSM_LV2 = 4096;
3844 0 literal RABSM_LOA = 8192;
3845 0 literal RABSM_LIM = 16384;
3846 0 literal RABSM_LOC = 65536;
3847 0 literal RABSM_WAT = 131072;
3848 0 literal RABSM_ULK = 262144;
3849 0 literal RABSM_RLK = 524288;
3850 0 literal RABSM_NLK = 1048576;
3851 0 literal RABSM_KGE = 2097152;
3852 0 literal RABSM_KGT = 4194304;
3853 0 literal RABSM_NXR = 8388608;
3854 0 literal RABSM_RNE = 16777216;
3855 0 literal RABSM_TMO = 33554432;
3856 0 literal RABSM_CVT = 67108864;
3857 0 literal RABSM_RNF = 134217728;
3858 0 literal RABSM_ETO = 268435456;
3859 0 literal RABSM_PTA = 536870912;

```

```

3860 0 literal RABSM_PMT = 1073741824;
3861 0 literal RABSM_CCO = -2147483648;
3862 0 literal RABSC_SEQ = 0;
3863 0 literal RABSC_KEY = 1;
3864 0 literal RABSC_RFA = 2;
3865 0 literal RABSC_STM = 3;
3866 0 literal RABSK_BLN = 68;
3867 0 literal RABSC_BLN = 68;
3868 0 literal RABSS_RABDEF = 68;
3869 0 macro RABSB_BID = 0,0,8,0 %;
3870 0 macro RABSB_BLN = 1,0,8,0 %;
3871 0 macro RABSR_ISI_OVERLAY = 2,0,16,0 %;
3872 0 macro RABSW_ISI = 2,0,16,0 %;
3873 0 ! (if in tab)
3874 0 macro RABSR_ISI_BITS = 2,0,16,0 %;
3875 0 macro RABSV_PPF_RAT = 2,6,8,0 %;
3876 0 literal RABSS_PPF_RAT = 8;
3877 0 macro RABSV_PPF_IND = 2,14,1,0 %;
3878 0 ! (i.e., restricted operations)
3879 0 macro RABSR_ROP_OVERLAY = 4,0,32,0 %;
3880 0 macro RABSL_ROP = 4,0,32,0 %;
3881 0 macro RABSR_ROP_BITS0 = 4,0,32,0 %;
3882 0 macro RABSV_ASY = 4,0,1,0 %;
3883 0 macro RABSV_TPT = 4,1,1,0 %;
3884 0 macro RABSV_REA = 4,2,1,0 %;
3885 0 macro RABSV_RRL = 4,3,1,0 %;
3886 0 macro RABSV_UIF = 4,4,1,0 %;
3887 0 macro RABSV_MAS = 4,5,1,0 %;
3888 0 macro RABSV_FDL = 4,6,1,0 %;
3889 0 macro RABSV_HSH = 4,7,1,0 %;
3890 0 macro RABSV_EOF = 4,8,1,0 %;
3891 0 macro RABSV_RAH = 4,9,1,0 %;
3892 0 macro RABSV_WBH = 4,10,1,0 %;
3893 0 macro RABSV_BIO = 4,11,1,0 %;
3894 0 macro RABSV_LV2 = 4,12,1,0 %;
3895 0 macro RABSV_LOA = 4,13,1,0 %;
3896 0 macro RABSV_LIM = 4,14,1,0 %;
3897 0
3898 0 the following bits are input to
3899 0 $find or $get, (see above also REA and RRL)
3900 0 (separate byte)
3901 0
3902 0 macro RABSV_LOC = 4,16,1,0 %;
3903 0 macro RABSV_WAT = 4,17,1,0 %;
3904 0 macro RABSV_ULK = 4,18,1,0 %;
3905 0 macro RABSV_RLK = 4,19,1,0 %;
3906 0 macro RABSV_NLK = 4,20,1,0 %;
3907 0 macro RABSV_KGE = 4,21,1,0 %;
3908 0 macro RABSV_KGT = 4,22,1,0 %;
3909 0 macro RABSV_NXR = 4,23,1,0 %;
3910 0 macro RABSV_RNE = 4,24,1,0 %;
3911 0 macro RABSV_TMO = 4,25,1,0 %;
3912 0 macro RABSV_CVT = 4,26,1,0 %;
3913 0 macro RABSV_RNF = 4,27,1,0 %;
3914 0 macro RABSV_ETO = 4,28,1,0 %;
3915 0 macro RABSV_PTA = 4,29,1,0 %;
3916 0 macro RABSV_PMT = 4,30,1,0 %;

```

sequential access
keyed access
rfa access
stream access (valid only for sequential org)
length of rab
length of rab
block id
block length
internal stream index
rat value for process-permanent files
indirect access to process-permanent file
record options
asynchronous operations
truncate put - allow sequential put not at
lock record for read only, allow other readers
read record regardless of lock
update if existent
mass-insert mode
fast record deletion
use hash code in bkt
connect to eof
read ahead
write behind
connect for bio only
level 2 RU lock consistency
use bucket fill percentage
compare for key limit reached on \$get/\$find seq. (idx only)

use locate mode
wait if record not available
manual unlocking
allow readers for this locked record
do not lock record
key > or =
key greater than
get non-existent record
read no echo
use time-out period
convert to upper case
read no filter
extended terminal operation
purge type ahead
use prompt buffer

```

3917 0 macro RAB$V_CCO = 4,31,1,0 %; ! cancel control o on output
3918 0
3919 0 the following bits are terminal qualifiers only
3920 0 (separate byte)
3921 0
3922 0
3923 0
3924 0 eof, thus truncating file (seq. org only)
3925 0
3926 0 these next two should be in the byte for bits
3927 0 input to $find or $get, but there is no room there
3928 0
3929 0 the following bits may be
3930 0 input to various rab-related
3931 0 operations
3932 0
3933 0 macro RAB$R_ROP_FIELDS = 4,0,32,0 %;
3934 0 macro RAB$B_ROPT = 5,0,8,0 %; ! various options
3935 0 macro RAB$B_ROP2 = 6,0,8,0 %; ! get/find options (use of this field discouraged
3936 0 ! due to REA and RRL being in a different byte)
3937 0 macro RAB$B_ROP3 = 7,0,8,0 %; ! terminal read options
3938 0
3939 0 macro RAB$L_STS = 8,0,32,0 %; ! status
3940 0 macro RAB$R_STV_OVERLAY = 12,0,32,0 %;
3941 0 macro RAB$L_STV = 12,0,32,0 %; ! status value
3942 0 macro RAB$R_STV_FIELDS = 12,0,32,0 %;
3943 0 macro RAB$W_STV0 = 12,0,16,0 %; ! low word of stv
3944 0 macro RAB$W_STV2 = 14,0,16,0 %; ! high word of stv
3945 0 macro RAB$R_RFA_OVERLAY = 16,0,0,0 %;
3946 0 macro RAB$W_RFA = 16,0,0,0 %;
3947 0 literal RAB$S_RFA = 6; ! record's file address
3948 0 macro RAB$R_RFA_FIELDS = 16,0,0,0 %;
3949 0 macro RAB$L_RFA0 = 16,0,32,0 %;
3950 0 macro RAB$W_RFA4 = 20,0,16,0 %;
3951 0 ! to the rfa field to be a move quad, overwriting
3952 0 ! this reserved word)
3953 0 macro RAB$L_CTX = 24,0,32,0 %; ! user context
3954 0 ! -----
3955 0 macro RAB$F_RAC = 30,0,8,0 %; ! record access
3956 0 macro RAB$B_TMO = 31,0,8,0 %; ! time-out period
3957 0 macro RAB$W_USZ = 32,0,16,0 %; ! user buffer size
3958 0 macro RAB$W_RSZ = 34,0,16,0 %; ! record buffer size
3959 0 macro RAB$L_UBF = 36,0,32,0 %; ! user buffer address
3960 0 macro RAB$L_RBF = 40,0,32,0 %; ! record buffer address
3961 0 macro RAB$L_RHB = 44,0,32,0 %; ! record header buffer addr
3962 0 macro RAB$R_KBF_OVERLAY = 48,0,32,0 %;
3963 0 macro RAB$L_KBF = 48,0,32,0 %; ! key buffer address
3964 0 macro RAB$L_PBF = 48,0,32,0 %; ! prompt buffer addr
3965 0 macro RAB$R_KSZ_OVERLAY = 52,0,8,0 %;
3966 0 macro RAB$B_KSZ = 52,0,8,0 %; ! key buffer size
3967 0 macro RAB$B_PSZ = 52,0,8,0 %; ! prompt buffer size
3968 0 macro RAB$B_KRF = 53,0,8,0 %; ! key of reference
3969 0 macro RAB$B_MBF = 54,0,8,1 %; ! multi-buffer count
3970 0 macro RAB$B_MBC = 55,0,8,0 %; ! multi-block count
3971 0 macro RAB$R_BKT_OVERLAY = 56,0,32,0 %;
3972 0 macro RAB$L_BKT = 56,0,32,0 %; ! bucket hash code, vbn, or rrn
3973 0 macro RAB$L_DCT = 56,0,32,0 %; ! duplicates count on key accessed on alternate key

```

```

3974 0 macro RAB$FAB = 60,0,32,0 %;      ! related 'ab for connect
3975 0 macro RAB$L_XAB = 64,0,32,0 %;      ! XAB address
3976 0
3977 0 *** MODULE $NAMDEF ***
3978 0
3979 0     name block field definitions
3980 0
3981 0     the nam block is used to communicate optional
3982 0     filename-related information
3983 0
3984 0 literal NAM$C_BID = 2;                  ! code for nam block
3985 0 literal NAM$C_MAXRSS = 255;             ! maximum resultant name string size (network)
3986 0 literal NAM$C_MAXRSSLCL = 255;         ! maximum resultant name string size (local)
3987 0 literal NAM$M_PWD = 1;
3988 0 literal NAM$M_FILL_1 = 2;
3989 0 literal NAM$M_FILL_2 = 4;
3990 0 literal NAM$M_SYNCRK = 8;
3991 0 literal NAM$M_NOCONCEAL = 16;
3992 0 literal NAM$M_SLPARSE = 32;
3993 0 literal NAM$M_SRCHXABS = 64;
3994 0 literal NAM$C_UFS = 0;                 ! Unknown file system for remote file access or
3995 0     not applicable for local file access or
3996 0     not applicable for task-to-task operation
3997 0 literal NAM$C_RMS11 = 1;                 ! RMS-11
3998 0 literal NAM$C_RMS20 = 2;                 ! RMS-20
3999 0 literal NAM$C_RMS32 = 3;                 ! RMS-32
4000 0 literal NAM$C_FCS11 = 4;                 ! FCS-11
4001 0 literal NAM$C_RT11FS = 5;                 ! RT-11 file system
4002 0 literal NAM$C_TOPS20FS = 7;                 ! TOPS-20 file system
4003 0 literal NAM$C_TOPS10FS = 8;                 ! TOPS-10 file system
4004 0 literal NAM$C_RMS32S = 10;                 ! RMS-32 subset (e.g., VAXELAN)
4005 0 *****
4006 0     the following 3 fields must not be rearranged relative to each other
4007 0
4008 0 literal NAM$C_DVI = 16;                   ! length of dvi field
4009 0 *****
4010 0     the location of the following fields must not
4011 0     be changed due to their commonality with the fib
4012 0 literal NAM$M_IFI = 65536;
4013 0 literal NAM$M_SRCHNMF = 1073741824;
4014 0 literal NAM$M_SVCTX = -2147483648;
4015 0 literal NAM$K_BLN_V2 = 56;                 ! Version 2 name block length
4016 0 literal NAM$C_BLN_V2 = 56;                 ! Version 2 name block length
4017 0 literal NAM$M_EXP_VER = 1;
4018 0 literal NAM$M_EXP_TYPE = 2;
4019 0 literal NAM$M_EXP_NAME = 4;
4020 0 literal NAM$M_WILD_VER = 8;
4021 0 literal NAM$M_WILD_TYPE = 16;
4022 0 literal NAM$M_WILD_NAME = 32;
4023 0 literal NAM$M_EXP_DIR = 64;
4024 0 literal NAM$M_EXP_DEV = 128;
4025 0 literal NAM$M_WILDCARD = 256;
4026 0 literal NAM$M_SEARCH_LIST = 2048;
4027 0 literal NAM$M_CNCL_DEV = 4096;
4028 0 literal NAM$M_ROOT_DIR = 8192;
4029 0 literal NAM$M_LOWVER = 16384;
4030 0 literal NAM$M_HIGHVER = 32768;

```

```

4031 0 literal NAMSM_PPF = 65536;
4032 0 literal NAMSM_NODE = 131072;
4033 0 literal NAMSM_QUOTED = 262144;
4034 0 literal NAMSM_GRP_MBR = 524288;
4035 0 literal NAMSM_WILD_DIR = 1048576;
4036 0 literal NAMSM_DIR_CVLS = 14680064;
4037 0 literal NAMSM_WILD_UFD = 16777216;
4038 0 literal NAMSM_WILD_SFD1 = 33554432;
4039 0 literal NAMSM_WILD_SFD2 = 67108864;
4040 0 literal NAMSM_WILD_SFD3 = 134217728;
4041 0 literal NAMSM_WILD_SFD4 = 268435456;
4042 0 literal NAMSM_WILD_SFD5 = 536870912;
4043 0 literal NAMSM_WILD_SFD6 = 1073741824;
4044 0 literal NAMSM_WILD_SFD7 = -2147483648;
4045 0 literal NAMSM_WILD_GRP = 16777216;
4046 0 literal NAMSM_WILD_MBR = 33554432;
4047 0 literal NAMSK_BLN_DIRWC = 96;
4048 0 literal NAMSK_BLN_DIRWC = 96;
4049 0 literal NAMSK_BLN = 96;
4050 0 literal NAMSK_BLN = 96;
4051 0 literal NAMSS_NAMDEF = 96;
4052 0 macro NAMSB_BID = 0,0,8,0 %;
4053 0 macro NAMSB_BLN = 1,0,8,0 %;
4054 0 ++++++
4055 0 the following 3 fields must not be rearranged relative to each other
4056 0
4057 0 macro NAMSB_RSS = 2,0,8,0 %;
4058 0 macro NAMSB_RSL = 3,0,8,0 %;
4059 0 macro NAMSL_RSA = 4,0,32,0 %;
4060 0 -----
4061 0 macro NAMSR_NOP_OVERLAY = 8,0,8,0 %;
4062 0 macro NAMSB_NOP = 8,0,8,0 %;
4063 0 macro NAMSR_NOP_BITS = 8,0,8,0 %;
4064 0 macro NAMSV_PWD = 8,0,1,0 %;
4065 0 macro NAMSV_FILL_1 = 8,1,1,0 %;
4066 0 macro NAMSV_FILL_2 = 8,2,1,0 %;
4067 0 macro NAMSV_SYNCRK = 8,3,1,0 %;
4068 0 macro NAMSV_NOCONCEAL = 8,4,1,0 %;
4069 0 macro NAMSV_SLPARSE = 8,5,1,0 %;
4070 0 macro NAMSV_SRCHXABS = 8,6,1,0 %;
4071 0 network (not documented) -- used by directory.
4072 0 other task-specific data of the form /netacp data"
4073 0 (default is to mask out password from expanded and
4074 0 resultant name strings and to create a logical name
4075 0 whose equivalence string is the unaltered nodespec)
4076 0 macro NAMSB_RFS = 9,0,8,0 %;
4077 0 Note: This field is reserved for use by Digital
4078 0 macro NAMSB_ESS = 10,0,8,0 %;
4079 0 macro NAMSB_ESL = 11,0,8,0 %;
4080 0 macro NAMSL_ESA = 12,0,32,0 %;
4081 0 -----
4082 0 macro NAMSL_RLF = 16,0,32,0 %;
4083 0 macro NAMST_DVI = 20,0,0,0 %;
4084 0 literal NAMSS_DVI = 16;
4085 0 macro NAMSR_FID_OVERLAY = 36,0,0,0 %;
4086 0 macro NAMSW_FID = 36,0,0,0 %;
4087 0 literal NAMSS_FID = 6;

```

! Not documented optional length
! Not documented optional length
! Name block length
! Name block length

! block id
! block length

! resultant string area size
! resultant string length
! resultant string area address

! Name options

! Return password if present in nodespec string and any
! unused. (used to be undocumented ROD)
! unused. (used to be undocumented SOD)
! Only do syntax check on \$sparse operation
! Do not conceal device/root directory
! Parse search list (not documented) -- used by BACKUP.
! Fill in attached XABS on \$SEARCH operations over the

! network (not documented) -- used by directory.

! other task-specific data of the form /netacp data"

! (default is to mask out password from expanded and

! resultant name strings and to create a logical name

! whose equivalence string is the unaltered nodespec)

! Remote file system type (currently not documented)

! Note: This field is reserved for use by Digital

! expanded string area size

! expanded string length

! expanded string area address

! -----

! related file nam block addr

! device id

! file id

C 15
15-Sep-1984 22:56:00
15-Sep-1984 22:55:58

VAX-11 Bliss-32 V4.0-742
_S255\$DUA28:[RMS.OBJ]RMS.R32;1

Page 77
(9)

```
4088 0 macro NAMSR_FID_FIELDS = 36,0,0,0 %;
4089 0 macro NAMSW_FID_NUM = 36,0,16,0 %;
4090 0 macro NAMSW_FID_SEQ = 38,0,16,0 %;
4091 0 macro NAMSR_FID_RVN_OVERLAY = 40,0,16,0 %;
4092 0 macro NAMSW_FID_RVN = 40,0,16,0 %;
4093 0 macro NAMSR_FID_RVN_FIELDS = 40,0,16,0 %;
4094 0 macro NAMSB_FID_RVN = 40,0,8,0 %;
4095 0 macro NAMSB_FID_NMX = 41,0,8,0 %;
4096 0 macro NAMSR_DID_OVERLAY = 42,0,0,0 %;
4097 0 macro NAMSW_DID = 42,0,0,0 %;
4098 0 literal NAMSS_DID = 6;
4099 0 macro NAMSR_DID_FIELDS = 42,0,0,0 %;
4100 0 macro NAMSW_DID_NUM = 42,0,16,0 %;
4101 0 macro NAMSW_DID_SEQ = 44,0,16,0 %;
4102 0 macro NAMSR_DID_RVN_OVERLAY = 46,0,16,0 %;
4103 0 macro NAMSW_DID_RVN = 46,0,16,0 %;
4104 0 macro NAMSR_DID_RVN_FIELDS = 46,0,16,0 %;
4105 0 macro NAMSB_DID_RVN = 46,0,8,0 %;
4106 0 macro NAMSB_DID_NMX = 47,0,8,0 %;
4107 0 macro NAMSR_WCC_OVERLAY = 48,0,32,0 %;
4108 0 macro NAMSL_WCC = 48,0,32,0 %;
4109 0 macro NAMSR_WCC_BITS = 48,0,32,0 %;
4110 0 macro NAMSV_IFI = 48,16,1,0 %;
4111 0 macro NAMSV_SRCHNMF = 48,30,1,0 %;
4112 0 macro NAMSV_SVCTX = 48,31,1,0 %;
4113 0 macro NAMSR_FNB_OVERLAY = 52,0,32,0 %;
4114 0 macro NAMSL_FNB = 52,0,32,0 %;
4115 0 macro NAMSR_FNB_BITS0 = 52,0,24,0 %;
4116 0 macro NAMSV_EXP_VER = 52,0,1,0 %;
4117 0 macro NAMSV_EXP_TYPE = 52,1,1,0 %;
4118 0 macro NAMSV_EXP_NAME = 52,2,1,0 %;
4119 0 macro NAMSV_WILD_VER = 52,3,1,0 %;
4120 0 macro NAMSV_WILD_TYPE = 52,4,1,0 %;
4121 0 macro NAMSV_WILD_NAME = 52,5,1,0 %;
4122 0 macro NAMSV_EXP_DIR = 52,6,1,0 %;
4123 0 macro NAMSV_EXP_DEV = 52,7,1,0 %;
4124 0 macro NAMSV_WILDCARD = 52,8,1,0 %;
4125 0 macro NAMSV_SEARCH_LIST = 52,11,1,0 %;
4126 0 macro NAMSV_CNCL_DEV = 52,12,1,0 %;
4127 0 macro NAMSV_ROOT_DIR = 52,13,1,0 %;
4128 0 macro NAMSV_LOWER = 52,14,1,0 %;
4129 0 macro NAMSV_HIGHER = 52,15,1,0 %;
4130 0 macro NAMSV_PPF = 52,16,1,0 %;
4131 0 macro NAMSV_NODE = 52,17,1,0 %;
4132 0 macro NAMSV_QUOTED = 52,18,1,0 %;
4133 0 macro NAMSV_GRP_MBR = 52,19,1,0 %;
4134 0 macro NAMSV_WILD_DIR = 52,20,1,0 %;
4135 0 macro NAMSV_DIR_LVL = 52,21,3,0 %;
4136 0 literal NAMSS_DIR_LVL = 3;
4137 0
4138 0 (inclusive or of other wild card bits)
4139 0 macro NAMSR_FNB_BITS1 = 52,0,32,0 %;
4140 0 macro NAMSV_WILD_UFD = 52,24,1,0 %;
4141 0 macro NAMSV_WILD_SFD1 = 52,25,1,0 %;
4142 0 macro NAMSV_WILD_SFD2 = 52,26,1,0 %;
4143 0 macro NAMSV_WILD_SFD3 = 52,27,1,0 %;
4144 0 macro NAMSV_WILD_SFD4 = 52,28,1,0 %;
```

```
file number
sequence number
relative volume number
alternate format RVN
alternate format file number extension
directory id
file number
sequence number
relative volume number
alternate format RVN
alternate format file number extension
wild card context
the first word contains an IFI
no-more-files has been encountered on a search
save context across search calls
file name status bits
version was explicit
type was explicit
name was explicit
version contained a wild card
type contained a wild card
name contained a wild card
directory was explicit
device was explicit
filename string included a wild card
search list present
concealed device present
root directory present
lower numbered version(s) of file exist(s)
higher
process-permanent file referenced indirectly
filename specification included a nodename
filename spec included a quoted string
directory spec was of group-member format
directory spec included a wild card
number of directory levels (0=ufd only)
ufd included a wild card
sfd1 included a wild card
sfd2 included a wild card
sfd3 included a wild card
sfd4 included a wild card
```

```

4145 0 macro NAMS_V_WILD_SFDS = 52,29,1,0 %; ! sfd5 included a wild card
4146 0 macro NAMS_V_WILD_SFDS6 = 52,30,1,0 %; ! sfd6 included a wild card
4147 0 macro NAMS_V_WILD_SFDS7 = 52,31,1,0 %; ! sfd7 included a wild card
4148 0 macro NAMS_V_FNB_BITS2 = 52,0,32,0 %;
4149 0 macro NAMS_V_WILD_GRP = 52,24,1,0 %; ! group contained a wild card
4150 0 macro NAMS_V_WILD_MBR = 52,25,1,0 %; ! member contained a wild card
4151 0 -----
4152 0 (prior to 40 byte extension)
4153 0
4154 0 Extend the NAM block by 40 bytes.
4155 0
4156 0 macro NAMS_B_NODE = 56,0,8,0 %; ! Nodespec length
4157 0 macro NAMS_B_DEV = 57,0,8,0 %; ! Device length
4158 0 macro NAMS_B_DIR = 58,0,8,0 %; ! Directory length
4159 0 macro NAMS_B_NAME = 59,0,8,0 %; ! Filename length
4160 0 macro NAMS_B_TYPE = 60,0,8,0 %; ! Filetype length
4161 0 macro NAMS_B_VER = 61,0,8,0 %; ! Version number length
4162 0 macro NAMS_L_NODE = 64,0,32,0 %; ! Nodespec address
4163 0 macro NAMS_L_DEV = 68,0,32,0 %; ! Device address
4164 0 macro NAMS_L_DIR = 72,0,32,0 %; ! Directory address
4165 0 macro NAMS_L_NAME = 76,0,32,0 %; ! Filename address
4166 0 macro NAMS_L_TYPE = 80,0,32,0 %; ! Filetype address
4167 0 macro NAMS_L_VER = 84,0,32,0 %; ! Version number address
4168 0
4169 0 *** MODULE $XABDEF ***
4170 0
4171 0 definitions for all xabs
4172 0 $xabdef
4173 0
4174 0
4175 0
4176 0 the first four fields are shared in common between all xabs
4177 0 and hence are defined only once
4178 0 (the only exception is that the spare word may be used by some xabs)
4179 0
4180 0 literal XABSS_XABDEF = 20;
4181 0 macro XABSS_COD = 0,0,8,0 %; ! xab id code
4182 0 macro XABSS_BLN = 1,0,8,0 %; ! block length
4183 0 macro XABSS_NXT = 4,0,32,0 %; ! xab chain link
4184 0 ! WITH POSSIBLE EXCEPTION OF SPARE FIELD
4185 0 macro XABSS_RVN = 8,0,16,0 %;
4186 0 macro XABSS_RDT_OVERLAY = 12,0,0,0 %;
4187 0 macro XABSS_RDT = 12,0,0,0 %;
4188 0 literal XABSS_RDT = 8;
4189 0 macro XABSS_RDT_FIELDS = 12,0,0,0 %;
4190 0 macro XABSS_RDT0 = 12,0,32,0 %;
4191 0 macro XABSS_RDT4 = 16,0,32,1 %;
4192 0 ! COMMON AMONG DAT AND RDT XABS
4193 0 literal XABSS_XABDEF1 = 23;
4194 0 macro XABSS_BRZ = 22,0,8,0 %; ! COMMON TO FHC AND ALQ XABS
4195 0 literal XABSS_CXT_VER1 = 1; ! RMS Context Extraction version 1
4196 0
4197 0 *** MODULE $XABFHCDEF ***
4198 0 ++
4199 0 file header characteristics xab definitions
4200 0 $xabfhcdef
4201 0

```



```

4202 0  | *****
4203 0  | the fields of this xab cannot be rearranged since
4204 0  | they correspond to an on-disk structure
4205 0  |
4206 0  | literal XABSC_FHC = 29;          | xabfhc id code
4207 0  | literal XABSK_FHCLEN = 44;       | length of xabfhc
4208 0  | literal XABSC_FHCLEN = 44;       | length of xabfhc
4209 0  | literal XABSS_XABFHCDEF = 44;
4210 0  | THESE 4 FIELDS ARE COMMON TO ALL XABS AND
4211 0  | HAVE BEEN DEFINED BY $XABDEF
4212 0  | macro XABSB_RFO = 8,0,8,0 %;     | record format and file org
4213 0  | macro XABSB_ATR = 9,0,8,0 %;     | record attributes
4214 0  | macro XABSW_LRL = 10,0,16,0 %;   | longest record's length
4215 0  | macro XABSR_HBK_OVERLAY = 12,0,32,0 %;
4216 0  | macro XABSL_HBK = 12,0,32,0 %;   | hi vbn allocated
4217 0  | ! (n.b. reversed on disk!)
4218 0  | macro XABSR_HBK_FIELDS = 12,0,32,0 %;
4219 0  | macro XABSW_HBK0 = 12,0,16,0 %;
4220 0  | macro XABSW_HBK2 = 14,0,16,0 %;
4221 0  | macro XABSR_EBK_OVERLAY = 16,0,32,0 %;
4222 0  | macro XABSL_EBK = 16,0,32,0 %;   | eof vbn
4223 0  | ! (n.b. reversed on disk)
4224 0  | macro XABSR_EBK_FIELDS = 16,0,32,0 %;
4225 0  | macro XABSW_EBK0 = 16,0,16,0 %;
4226 0  | macro XABSW_EBK2 = 18,0,16,0 %;
4227 0  | macro XABSW_FFB = 20,0,16,0 %;   | first free byte in eof block
4228 0  | ! defined above in $xabdef, since it is shared
4229 0  | ! by the all xab)
4230 0  | macro XABSB_HSZ = 23,0,8,0 %;    | header size for vfc
4231 0  | macro XABSW_MRZ = 24,0,16,0 %;   | max record size
4232 0  | macro XABSW_DXQ = 26,0,16,0 %;   | default extend quantity
4233 0  | macro XABSW_GBC = 28,0,16,0 %;   | global buffer count
4234 0  | macro XABSW_VERLIMIT = 38,0,16,0 %; | version limit for file.
4235 0  | -----
4236 0  | macro XABSL_SBN = 40,0,32,0 %;   | starting lbn if contiguous
4237 0  |
4238 0  | *** MODULE $XABALLDEF ***
4239 0  | --
4240 0  | ++
4241 0  |
4242 0  | allocation xab definitions
4243 0  | $xaballdef
4244 0  |
4245 0  |
4246 0  | *****
4247 0  | the fields thru bkz cannot be rearranged due to
4248 0  | their commonality with fab
4249 0  | literal XABSC_ALL = 20;          | xaball id code
4250 0  | literal XABSM_HRD = 1;
4251 0  | literal XABSM_ONC = 2;
4252 0  | literal XABSM_CBT = 32;
4253 0  | literal XABSM_CTG = 128;
4254 0  | literal XABSC_ANY = 0;           | any allocation o.k.
4255 0  | literal XABSC_CYL = 1;           | cylinder boundary
4256 0  | literal XABSC_LBN = 2;           | allocate at specified lbn
4257 0  | literal XABSC_VBN = 3;           | allocate near specified vbn
4258 0  | literal XABSC_RFI = 4;           | allocate near related file

```

```

4259 0 literal XAB$K_ALLLEN = 32;          ! length of xaball
4260 0 literal XAB$C_ALLLEN = 32;          ! length of xaball
4261 0 literal XAB$S_XABALLDEF = 32;
4262 0 ! THESE 4 FIELDS ARE COMMON TO ALL XABS AND
4263 0 ! HAVE BEEN DEFINED BY $XABDEF
4264 0 macro XAB$R_AOP_OVERLAY = 8,0,8,0 %;
4265 0 macro XAB$B_AOP = 8,0,8,0 %;          ! allocation options
4266 0 macro XAB$R_AOP_BITS = 8,0,8,0 %;
4267 0 macro XAB$V_HRD = 8,0,1,0 %;          ! fail if requested alignment impossible
4268 0 macro XAB$V_ONC = 8,1,1,0 %;          ! locate allocated space within a cylinder
4269 0 macro XAB$V_CBT = 8,5,1,0 %;          ! contiguous allocation, best try
4270 0 macro XAB$V_CTG = 8,7,1,0 %;          ! contiguous allocation
4271 0 macro XAB$B_ALN = 9,0,8,0 %;          ! alignment type
4272 0 macro XAB$W_VOL = 10,0,16,0 %;        ! relative volume no. for allocation
4273 0 ! (not applicable if aln = vbn or rfi)
4274 0 macro XAB$L_LOC = 12,0,32,0 %;        ! allocation location
4275 0 macro XAB$L_ALQ = 16,0,32,0 %;        ! allocation quantity
4276 0 macro XAB$W_DEQ = 20,0,16,0 %;        ! default allocation quantity
4277 0 ! defined above in $xabdef, since it is shared by the fhc
4278 0 ! xab and has the same offset, of course)
4279 0 -----*****
4280 0 macro XAB$B_AID = 23,0,8,0 %;          ! area id number
4281 0 macro XAB$R_RFI_OVERLAY = 24,0,0,0 %;
4282 0 macro XAB$W_RFI = 24,0,0,0 %;
4283 0 literal XAB$S_RFI = 6;                ! related file id
4284 0 macro XAB$R_RFI_FIELDS = 24,0,0,0 %;
4285 0 macro XAB$W_RFI0 = 24,0,16,0 %;        ! file number
4286 0 macro XAB$W_RFI2 = 26,0,16,0 %;        ! seq number
4287 0 macro XAB$W_RFI4 = 28,0,16,0 %;        ! rev number
4288 0
4289 0 *** MODULE $XABDATDEF ***
4290 0 --
4291 0 ++
4292 0
4293 0 date/time xab definitions
4294 0 $xabdatdef
4295 0
4296 0 literal XAB$C_DAT = 18;                ! xabdat id code
4297 0 literal XAB$K_DATLEN_V2 = 36;          ! Version 2 XABDAT length
4298 0 literal XAB$C_DATLEN_V2 = 36;          ! Version 2 XABDAT length
4299 0 literal XAB$K_DATLEN = 44;             ! length of XABDAT
4300 0 literal XAB$C_DATLEN = 44;             ! length of XABDAT
4301 0 literal XAB$S_XABDATDEF = 44;
4302 0 ! THESE 4 FIELDS ARE COMMON TO ALL XABS AND
4303 0 ! HAVE BEEN DEFINED BY $XABDEF
4304 0 macro XAB$R_CDT_OVERLAY = 20,0,0,0 %;
4305 0 macro XAB$Q_CDT = 20,0,0,0 %;
4306 0 literal XAB$S_CDT = 8;                ! creation date & time
4307 0 macro XAB$R_CDT_FIELDS = 20,0,0,0 %;
4308 0 macro XAB$L_CDT0 = 20,0,32,0 %;
4309 0 macro XAB$L_CDT4 = 24,0,32,1 %;
4310 0 macro XAB$R_EDT_OVERLAY = 28,0,0,0 %;
4311 0 macro XAB$Q_EDT = 28,0,0,0 %;
4312 0 literal XAB$S_EDT = 8;                ! expiration date & time
4313 0 macro XAB$R_EDT_FIELDS = 28,0,0,0 %;
4314 0 macro XAB$L_EDT0 = 28,0,32,0 %;
4315 0 macro XAB$L_EDT4 = 32,0,32,1 %;

```

```

4316 0 macro XAB$R_BDT_OVERLAY = 36,0,0,0 %;
4317 0 macro XAB$Q_BDT = 36,0,0,0 %;
4318 0 literal XAB$S_BDT = 8; ! backup date and time
4319 0 macro XAB$R_BDT_FIELDS = 36,0,0,0 %;
4320 0 macro XAB$L_BDT0 = 36,0,32,0 %;
4321 0 macro XAB$L_BDT4 = 40,0,32,1 %;
4322 0
4323 0 *** MODULE $XABRDTDEF ***
4324 0 --
4325 0 ++
4326 0
4327 0 revision date/time xab definitions
4328 0 $xabrtdtdef
4329 0 literal XAB$C_RDT = 30; ! xabrdt id code
4330 0 literal XAB$K_RDTLEN = 20; ! length of rdt xab
4331 0 literal XAB$C_RDTLEN = 20; ! length of rdt xab
4332 0 literal XAB$S_XABRDTDEF = 20;
4333 0 ! THESE 4 FIELDS ARE COMMON TO ALL XABS AND
4334 0 ! HAVE BEEN DEFINED BY $XABDEF
4335 0
4336 0 *** MODULE $XABPRODEF ***
4337 0 --
4338 0 ++
4339 0
4340 0 protection xab field definitions
4341 0 $xabprodef
4342 0
4343 0
4344 0 literal XAB$C_PRO = 19; ! xabpro id code
4345 0 literal XAB$M_NOREAD = 1;
4346 0 literal XAB$M_NOWRITE = 2;
4347 0 literal XAB$M_NOEXE = 4;
4348 0 literal XAB$M_NODEL = 8;
4349 0 literal XAB$S_XABPRODEF = 1;
4350 0 macro XAB$R_XABPRODEF BITS = 0,0,8,0 %;
4351 0 macro XAB$V_NOREAD = 0,0,1,0 %; ! deny read access
4352 0 macro XAB$V_NOWRITE = 0,1,1,0 %; ! deny write access
4353 0 macro XAB$V_NOEXE = 0,2,1,0 %; ! deny execution access
4354 0 macro XAB$V_NODEL = 0,3,1,0 %; ! deny delete access
4355 0 literal XAB$M_PROPAGATE = 1;
4356 0 literal XAB$K_PROLEN_V3 = 16; ! V3a xabpro length
4357 0 literal XAB$C_PROLEN_V3 = 16; ! V3a xabpro length
4358 0 literal XAB$K_PROLEN = 88; ! xabpro length
4359 0 literal XAB$C_PROLEN = 88; ! xabpro length
4360 0 literal XAB$S_XABPRODEF1 = 88;
4361 0 ! THESE 4 FIELDS ARE COMMON TO ALL XABS AND
4362 0 ! HAVE BEEN DEFINED BY $XABDEF
4363 0 macro XAB$R_PRO_OVERLAY = 8,0,16,0 %;
4364 0 macro XAB$W_PRO = 8,0,16,0 %; ! protection mask
4365 0 macro XAB$R_PRO_BITS = 8,0,16,0 %;
4366 0 macro XAB$V_SYS = 8,0,4,0 %;
4367 0 literal XAB$S_SYS = 4; ! system
4368 0 macro XAB$V_OWN = 8,4,4,0 %;
4369 0 literal XAB$S_OWN = 4; ! owner
4370 0 macro XAB$V_GRP = 8,8,4,0 %;
4371 0 literal XAB$S_GRP = 4; ! group
4372 0 macro XAB$V_WED = 8,12,4,0 %;

```

```

4373 0 literal XAB$S_WLD = 4; | world
4374 0 macro XAB$B_MTACC = 10,0,8,0 %; | Magtape access control char.
4375 0 macro XAB$R_PROT_OPT_OVERLAY = 11,0,8,0 %; |
4376 0 macro XAB$B_PROT_OPT = 11,0,8,0 %; | XABPRO options field
4377 0 macro XAB$R_PROT_OPT_FIELDS = 11,0,8,0 %; |
4378 0 macro XAB$V_PROPAGATE = 11,0,1,0 %; | Propagate security attributes on $ENTER and $RENAME
4379 0 macro XAB$R_UIC_OVERLAY = 12,0,32,0 %; |
4380 0 macro XAB$L_UIC = 12,0,32,0 %; | uic code
4381 0 macro XAB$R_UIC_FIELDS = 12,0,32,0 %; |
4382 0 macro XAB$W_MBM = 12,0,16,0 %; | member code
4383 0 macro XAB$W_GRP = 14,0,16,0 %; | group code
4384 0 macro XAB$R_PROT_MODE_OVERLAY = 16,0,0,0 %; | ! RWED/mode protection for file
4385 0 macro XAB$Q_PROT_MODE = 16,0,0,0 %; |
4386 0 literal XAB$S_PROT_MODE = 8; | eventually may be a quadword
4387 0 macro XAB$R_PROT_MODE_FIELDS = 16,0,8,0 %; |
4388 0 macro XAB$B_PROT_MODE = 16,0,8,0 %; | but currently only a byte
4389 0 macro XAB$L_ACLBUF = 24,0,32,0 %; | address of user's ACL buffer
4390 0 macro XAB$W_ACLSZ = 28,0,16,0 %; | size of user's ACL buffer
4391 0 macro XAB$W_ACLLEN = 30,0,16,0 %; | return length of entire ACL
4392 0 macro XAB$L_ACLCTX = 32,0,32,0 %; | ACL context field
4393 0 macro XAB$L_ACLSTS = 36,0,32,0 %; | ACL return err status
4394 0
4395 0 |*** MODULE $XABTRMDEF ***
4396 0 |--
4397 0 |++
4398 0
4399 0 | terminal control xab field definitions
4400 0 | $xabtrmdef
4401 0 |
4402 0 |
4403 0 |
4404 0 | literal XAB$C_TRM = 31; | XABTRM ID CODE
4405 0 | literal XAB$K_TRMLEN = 36; | length of xab of type terminal control
4406 0 | literal XAB$C_TRMLEN = 36; | length of xab of type terminal control
4407 0 | literal XAB$S_XABTRMDEF = 36; |
4408 0 | ! THESE 4 FIELDS ARE COMMON TO ALL XABS AND
4409 0 | ! HAVE BEEN DEFINED BY $XABDEF
4410 0 | macro XAB$L_ITMLST = 8,0,32,0 %; | item list address
4411 0 | macro XAB$W_ITMLST_LEN = 12,0,16,0 %; | item list length
4412 0 |
4413 0 |*** MODULE $XABSUMDEF ***
4414 0 |--
4415 0 |++
4416 0 |
4417 0 | summary xab field definitions
4418 0 | $xabsumdef
4419 0 |
4420 0 |
4421 0 | literal XAB$C_SUM = 22; | xabsum id code
4422 0 | literal XAB$K_SUMLN = 12; | xabsum length
4423 0 | literal XAB$C_SUMLN = 12; | xabsum length
4424 0 | literal XAB$S_XABSUMDEF = 12; |
4425 0 | ! THESE 4 FIELDS ARE COMMON TO ALL XABS AND
4426 0 | ! HAVE BEEN DEFINED BY $XABDEF
4427 0 | macro XAB$B_NOA = 8,0,8,0 %; | number of defined areas for index file
4428 0 | macro XAB$B_NOK = 9,0,8,0 %; | number of defined keys for index file
4429 0 | macro XAB$W_PVN = 10,0,16,0 %; | prologue version number (relative and index files)

```

```

4430 0
4431 00 *** MODULE $XABKEYDEF ***
4432 00 --
4433 00 ++
4434 00
4435 00 key definition xab field definitions
4436 00 $xabkeydef
4437 00
4438 00
4439 00 literal XABSC_KEY = 21; . xabkey id code
4440 00 literal XABSM_DUP = 1;
4441 00 literal XABSM_CHG = 2;
4442 00 literal XABSM_NUL = 4;
4443 00 literal XABSM_IDX_NCMPR = 8;
4444 00 literal XABSM_KEY_NCMPR = 64;
4445 00 literal XABSM_DAT_NCMPR = 128;
4446 00 literal XABSC_STG = 0; ! string
4447 00 literal XABSC_IN2 = 1; ! signed 15 bit integer (2 bytes)
4448 00 literal XABSC_BN2 = 2; ! 2 byte binary
4449 00 literal XABSC_IN4 = 3; ! signed 31 bit integer (4 bytes)
4450 00 literal XABSC_BN4 = 4; ! 4 byte binary
4451 00 literal XABSC_PAC = 5; ! packed decimal (1-16 bytes)
4452 00 literal XABSC_IN8 = 6; ! signed 63 bit integer (4 bytes)
4453 00 literal XABSC_BN8 = 7; ! 8 byte binary
4454 00 literal XABSC_MAXDTP = 7; ! max. legal data type
4455 00 literal XABSK_KEYLEN_V2 = 64; ! old xabkey length
4456 00 literal XABSC_KEYLEN_V2 = 64; ! old xabkey length
4457 00
4458 00 Additions for prologue 3 files
4459 00
4460 00 literal XABSC_PRG3 = 3; ! Prologue version three
4461 00 literal XABSC_PRG2 = 2; ! Prologue version two
4462 00 literal XABSC_PRG1 = 1; ! Prologue versoin one
4463 00 literal XABSK_KEYLEN = 76; ! xabkey length
4464 00 literal XABSC_KEYLEN = 76; ! xabkey length
4465 00 --
4466 00 ++
4467 00 literal XABSS_XABKEYDEF = 76;
4468 00 THESE 4 FIELDS ARE COMMON TO ALL XABS AND
4469 00 HAVE BEEN DEFINED BY $XABDEF
4470 00
4471 00 the field layout of the key xab is such that it matchs as
4472 00 closely as possible the layout of a key decriptor in the
4473 00 index file prologue. this is so the contents may be moved
4474 00 between the two structures as efficiently as possible.
4475 00
4476 00 macro XABSB_IAN = 8,0,8,0 %; ! index level area number
4477 00 macro XABSB_LAN = 9,0,8,0 %; ! lowest index level area number
4478 00 macro XABSB_DAN = 10,0,8,0 %; ! data level area number
4479 00 macro XABSB_LVL = 11,0,8,0 %; ! level of root bucket
4480 00 macro XABSB_IBS = 12,0,8,0 %; ! size of index buckets in virtual blocks
4481 00 macro XABSB_DBS = 13,0,8,0 %; ! size of data buckets in virtual blocks
4482 00 macro XABSL_RVB = 14,0,32,0 %; ! root bucket start vbn
4483 00 macro XABSR_FLG_OVERLAY = 18,0,8,0 %;
4484 00 macro XABSB_FLG = 18,0,8,0 %; ! key option flags
4485 00 macro XABSR_FLG_BITS0 = 18,0,8,0 %;
4486 00 macro XABSV_DUP = 18,0,1,0 %; ! duplicate key values allowed

```

```

4487 0 macro XAB$V_CHG = 18,1,1,0 %;
4488 0 macro XAB$V_NUL = 18,2,1,0 %;
4489 0 macro XAB$V_IDX_NCMPR = 18,3,1,0 %;
4490 0 macro XAB$V_KEY_NCMPR = 18,6,1,0 %;
4491 0 macro XAB$R_FLG_BITS1 = 18,0,8,0 %;
4492 0 macro XAB$V_DAT_NCMPR = 18,7,1,0 %;
4493 0 macro XAB$B_DTP = 19,0,8,0 %;
4494 0 macro XAB$B_NSG = 20,0,8,0 %;
4495 0 macro XAB$B_NUL = 21,0,8,0 %;
4496 0 macro XAB$B_TKS = 22,0,8,0 %;
4497 0 macro XAB$B_REF = 23,0,8,0 %;
4498 0 ! 1-254 = alternate keys)
4499 0 macro XAB$W_MRL = 24,0,16,0 %;
4500 0 macro XAB$W_IFL = 26,0,16,0 %;
4501 0 macro XAB$W_DFL = 28,0,16,0 %;
4502 0 macro XAB$R_POS_OVERLAY = 30,0,0,0 %;
4503 0 macro XAB$W_POS = 30,0,0,0 %;
4504 0 literal XAB$S_POS = 16;
4505 0 macro XAB$R_POS_FIELDS = 30,0,0,0 %;
4506 0 macro XAB$W_POS0 = 30,0,16,0 %;
4507 0 macro XAB$W_POS1 = 32,0,16,0 %;
4508 0 macro XAB$W_POS2 = 34,0,16,0 %;
4509 0 macro XAB$W_POS3 = 36,0,16,0 %;
4510 0 macro XAB$W_POS4 = 38,0,16,0 %;
4511 0 macro XAB$W_POS5 = 40,0,16,0 %;
4512 0 macro XAB$W_POS6 = 42,0,16,0 %;
4513 0 macro XAB$W_POS7 = 44,0,16,0 %;
4514 0 macro XAB$R_SIZ_OVERLAY = 46,0,0,0 %;
4515 0 macro XAB$B_SIZ = 46,0,0,0 %;
4516 0 literal XAB$S_SIZ = 8;
4517 0 macro XAB$R_SIZ_FIELDS = 46,0,0,0 %;
4518 0 macro XAB$B_SIZ0 = 46,0,8,0 %;
4519 0 macro XAB$B_SIZ1 = 47,0,8,0 %;
4520 0 macro XAB$B_SIZ2 = 48,0,8,0 %;
4521 0 macro XAB$B_SIZ3 = 49,0,8,0 %;
4522 0 macro XAB$B_SIZ4 = 50,0,8,0 %;
4523 0 macro XAB$B_SIZ5 = 51,0,8,0 %;
4524 0 macro XAB$B_SIZ6 = 52,0,8,0 %;
4525 0 macro XAB$B_SIZ7 = 53,0,8,0 %;
4526 0
4527 0 ! the positions of the above fields are dictated by the key descriptor
4528 0 ! record layout in the index file prologue.
4529 0
4530 0 macro XAB$L_KNM = 56,0,32,0 %;
4531 0 macro XAB$L_DVB = 60,0,32,0 %;
4532 0 macro XAB$R_TYP_OVERLAY = 64,0,0,0 %;
4533 0 macro XAB$B_TYP = 64,0,0,0 %;
4534 0 literal XAB$S_TYP = 8;
4535 0 macro XAB$R_TYP_FIELDS = 64,0,0,0 %;
4536 0 macro XAB$B_TYP0 = 64,0,8,0 %;
4537 0 macro XAB$B_TYP1 = 65,0,8,0 %;
4538 0 macro XAB$B_TYP2 = 66,0,8,0 %;
4539 0 macro XAB$B_TYP3 = 67,0,8,0 %;
4540 0 macro XAB$B_TYP4 = 68,0,8,0 %;
4541 0 macro XAB$B_TYP5 = 69,0,8,0 %;
4542 0 macro XAB$B_TYP6 = 70,0,8,0 %;
4543 0 macro XAB$B_TYP7 = 71,0,8,0 %;

```

```

! alt key only --key field may change on update
! alt key only --null key value enable
! indicate index records for given key are not compressed
! indicates key is not compressed in data record

! data record is not compressed
! key field data type
! number of key segments
! nul key character
! total key field size (bytes)
! key of reference (0=prim key,

! minimum record length to contain key field
! index bucket fill size (bytes)
! data bucket fil size (bytes)

! key field record offset positions

! segment 0
! segment 1
! segment 2
! segment 3
! segment 4
! segment 5
! segment 6
! segment 7

! key field segment sizes

! segment 0
! segment 1
! segment 2
! segment 3
! segment 4
! segment 5
! segment 6
! segment 7

! pointer to 32 character key name buffer
! first data bucket start vbn

! key field segment types

! segment 0
! segment 1
! segment 2
! segment 3
! segment 4
! segment 5
! segment 6
! segment 7

```

```

4544 0 macro XAB$B_PROLOG = 72,0,8,0 %;      ! indicate prologue version desired (primary key only)
4545 0
4546 0
4547 0
4548 0
4549 0
4550 0
4551 0
4552 0
4553 0
4554 0
4555 0
4556 0
4557 0
4558 0
4559 0
4560 0
4561 0
4562 0
4563 0
4564 0
4565 0
4566 0
4567 0
4568 0
4569 0
4570 0
4571 0
4572 0
4573 0
4574 0
4575 0
4576 0
4577 0
4578 0
4579 0
4580 0
4581 0
4582 0
4583 0
4584 0
4585 0
4586 0
4587 0
4588 0
4589 0
4590 0
4591 0
4592 0
4593 0
4594 0
4595 0
4596 0
4597 0
4598 0
4599 0
4600 0

```

```

*** MODULE $XABCXFDEF ***

RMS Context XAB associated with the FAB
  $xabcxfdef

Literal XAB$C_CXF = 32;      ! XABCXF id code
Literal XAB$M_CXFRST = 1;
Literal XAB$K_CXFLEN = 60;   ! length of xab type CXF
Literal XAB$C_CXFLEN = 60;   ! length of xab type CXF
Literal XAB$S_XABCXFDEF = 60;
  UP TILL NOW COMMON AMONG ALL XABS

Following in common with the CXR block, too.
Do not rearrange without changing both.

macro XAB$L_CXFSTS = 8,0,32,0 %;      ! Status of the last file operation.
macro XAB$L_CXFSTV = 12,0,32,0 %;     ! Status Value of the last file operation.

Top four bits of the options longword are reserved for the XABCXR. These
bits describe the version of the key buffer.

macro XAB$R_CXFCOP_OVERLAY = 16,0,32,0 %;
macro XAB$L_CXFCOP = 16,0,32,0 %;     ! Context Options.
macro XAB$R_CXFCOP_BITS = 16,0,32,0 %;
macro XAB$V_CXFRST = 16,0,32,0 %;     ! Restore file state - use context blk as input.
macro XAB$L_CXFBKP = 20,0,32,0 %;     ! Bookkeeping bits
macro XAB$W_CXFIFI = 24,0,16,0 %;     ! Internal File Identifier
macro XAB$B_CXFVER = 26,0,8,0 %;      ! prologue version num

Up Till now in common with XABCXR, too.

The following fields correspond to those in the FAB or IFB
They should not be rearranged as their order is assumed for
purposes of moving large chunks of data rather than a byte
or word at a time. Note: ASSUME is used in the actual code

macro XAB$W_CXFDEQ = 32,0,16,0 %;     ! Default extention quantity
macro XAB$B_CXFFAC = 34,0,8,0 %;      ! File access
macro XAB$B_CXFSHR = 35,0,8,0 %;      ! File Sharing
macro XAB$W_CXFRTS = 36,0,16,0 %;     ! (Not used)
macro XAB$B_CXFORG = 39,0,8,0 %;      ! file organization
macro XAB$W_CXFGBC = 40,0,16,0 %;     ! global buffer count
macro XAB$B_CXFRTV = 42,0,8,0 %;      ! retrieval window

*** MODULE $XABCXRDEF ***

RMS Context XAB associated with the RAB
  $xabcxrdef

Literal XAB$C_CXR = 33;      ! XABCXR id code
Literal XAB$M_CXRRST = 1;
Literal XAB$C_CXB_VER1 = 1;
Literal XAB$C_CXRLEN = 512;      ! Length of CXRBUF (bytes)

```

```

4601 0 literal XAB$K_CXRLEN = 84;          ! Length of XAB type CXR
4602 0 literal XAB$C_CXRLEN = 84;          ! Length of XAB type CXR
4603 0 literal XAB$C_CXRDEF = 84;
4604 0 UP TILL NOW COMMON AMONG ALL XABS
4605 0
4606 0         Following in common with the CXF block, too.
4607 0         Do not rearrange without changing it.
4608 0
4609 0 macro XAB$K_CXRSTS = 8,0,32,0 %;      ! Status of the last record operation.
4610 0 macro XAB$K_CXRSTV = 12,0,32,0 %;    ! Status Value of the last record operation.
4611 0 macro XAB$R_CXRCOP_OVERLAY = 16,0,32,0 %;
4612 0 macro XAB$R_CXRCOP = 16,0,32,0 %;    ! Context Options.
4613 0 macro XAB$R_CXRCOP_BITS = 16,0,32,0 %;
4614 0 macro XAB$V_CXRRST = 16,0,1,0 %;    ! Restore file/record state - use context blk as input.
4615 0 macro XAB$V_CXRBVER = 16,28,4,0 %;
4616 0 literal XAB$S_CXRBVER = 4;           ! Version of Key buffer
4617 0 macro XAB$K_CXRBKP = 20,0,32,0 %;    ! Bookkeeping bits
4618 0 macro XAB$W_CXRISI = 24,0,16,0 %;    ! Internal Record Identifier
4619 0 macro XAB$B_CXRVER = 26,0,8,0 %;     ! prologue version num.
4620 0
4621 0         Up Till now in common with XAB$CXF, too.
4622 0
4623 0
4624 0         The following elements are arranged such that large amounts of
4625 0         data can be moved at a time rather than words or bytes. Do not
4626 0         rearrange them without this consideration in mind.
4627 0
4628 0         The following elements are stream dependent regardless of file org.
4629 0
4630 0 macro XAB$B_CXRMBF = 32,0,8,0 %;        ! Multibuffer count
4631 0 macro XAB$B_CXRMBL = 33,0,8,0 %;      ! Multiblock count
4632 0 macro XAB$W_CXRBIZ = 34,0,16,0 %;    ! sz in byte of CXRBUF
4633 0
4634 0         The following elements are necessary for saving the NRP context for
4635 0         Sequential and Relative files.
4636 0
4637 0 macro XAB$K_CXRVCN = 36,0,32,0 %;      ! NRP VBN
4638 0 macro XAB$W_CXROFF = 40,0,16,0 %;    ! NRP offset in VBN
4639 0 macro XAB$W_CXR_FILL_8 = 42,0,16,0 %; ! mbz - longword align
4640 0
4641 0         The following elements are necessary for saving the NRP context for
4642 0         ISAM files.
4643 0
4644 0 macro XAB$K_CXRPOS0 = 44,0,32,0 %;      ! Primary Positioning RFA
4645 0 macro XAB$W_CXRPOS4 = 48,0,16,0 %;
4646 0 macro XAB$K_CXRCUR0 = 52,0,32,0 %;     ! Current Positioning RFA
4647 0 macro XAB$W_CXRCUR4 = 56,0,16,0 %;
4648 0 macro XAB$K_CXRSID0 = 60,0,32,0 %;     ! SIDR positioning PFA
4649 0 macro XAB$W_CXRSID4 = 64,0,16,0 %;
4650 0 macro XAB$W_CXRCNT = 68,0,16,0 %;      ! SIDR array count
4651 0 macro XAB$B_CXRKREF = 70,0,8,0 %;      ! Cur Key of Reference
4652 0 macro XAB$B_CXRKLEN = 71,0,8,0 %;      ! Length of key
4653 0 macro XAB$K_CXRBUF = 72,0,32,0 %;      ! address of key buf
4654 0
4655 0 *** MODULE $XABJNLDEF ***
4656 0 **
4657 0

```



```

4658 0      Journal XAB definitions
4659 0      $xabjnldef
4660 0
4661 0
4662 0      literal XAB$C_JNL = 34;          ! xabjnl id code
4663 0      literal XAB$M_ONLY_RU = 1;
4664 0      literal XAB$M_RU = 2;
4665 0      literal XAB$M_BI = 4;
4666 0      literal XAB$M_AI = 8;
4667 0      literal XAB$M_AT = 16;
4668 0      literal XAB$M_NEVER_RU = 32;
4669 0      literal XAB$K_MAXJNLNAM = 16;    ! max size of ascii string journal name
4670 0      literal XAB$C_MAXJNLNAM = 16;    ! max size of ascii string journal name
4671 0      literal XAB$K_JNLLEN = 60;
4672 0      literal XAB$C_JNLLEN = 60;
4673 0      literal XAB$S_XABJNLDEF = 60;
4674 0      ! THESE 4 FIELDS ARE COMMON TO ALL XABS AND
4675 0      ! HAVE BEEN DEFINED BY $XABDEF
4676 0      macro XAB$R_JOP_OVERLAY = 8,0,16,0 %;
4677 0      macro XAB$W_JOP = 8,0,16,0 %;      ! journaling flags
4678 0      macro XAB$R_JOP_BITS = 8,0,8,0 %;
4679 0      macro XAB$V_ONLY_RU = 8,0,1,0 %;    ! Recovery-unit only access
4680 0      macro XAB$V_RU = 8,1,1,0 %;        ! Recovery unit
4681 0      macro XAB$V_BI = 8,2,1,0 %;        ! Before Image
4682 0      macro XAB$V_AI = 8,3,1,0 %;        ! After Image
4683 0      macro XAB$V_AT = 8,4,1,0 %;        ! Audit Trail
4684 0      macro XAB$V_NEVER_RU = 8,5,1,0 %;   ! Never journal in Recovery-unit
4685 0      macro XAB$B_BIS = 12,0,8,0 %;      ! BI journal name buffer size
4686 0      macro XAB$B_BIL = 13,0,8,0 %;      ! BI journal name return size
4687 0      macro XAB$B_BIA = 16,0,32,0 %;     ! BI journal name buffer address
4688 0      macro XAB$B_AIS = 20,0,8,0 %;      ! AI journal name buffer size
4689 0      macro XAB$B_AIL = 21,0,8,0 %;      ! AI journal name return size
4690 0      macro XAB$B_AIA = 24,0,32,0 %;     ! AI journal name buffer address
4691 0      macro XAB$B_ATS = 28,0,8,0 %;      ! AT journal name buffer size
4692 0      macro XAB$B_ATL = 29,0,8,0 %;      ! AT journal name return size
4693 0      macro XAB$B_ATA = 32,0,32,0 %;     ! AT journal name buffer address
4694 0
4695 0      *** MODULE $FSCNDEF ***
4696 0      **
4697 0
4698 0      Descriptor codes for SYSS$FILESCAN
4699 0
4700 0
4701 0      literal FSCN$M_NODE = 1;
4702 0      literal FSCN$M_DEVICE = 2;
4703 0      literal FSCN$M_ROOT = 4;
4704 0      literal FSCN$M_DIRECTORY = 8;
4705 0      literal FSCN$M_NAME = 16;
4706 0      literal FSCN$M_TYPE = 32;
4707 0      literal FSCN$M_VERSION = 64;
4708 0      literal FSCN$S_FLD$FLAGS = 1;
4709 0      macro FSCN$V_NODE = 0,0,1,0 %;        ! Node name present
4710 0      macro FSCN$V_DEVICE = 0,1,1,0 %;     ! Device name present
4711 0      macro FSCN$V_ROOT = 0,2,1,0 %;        ! Root directory present
4712 0      macro FSCN$V_DIRECTORY = 0,3,1,0 %;   ! Directory present
4713 0      macro FSCN$V_NAME = 0,4,1,0 %;        ! File name present
4714 0      macro FSCN$V_TYPE = 0,5,1,0 %;        ! File type present

```

```

4715 0 macro FSCNSV VERSION = 0,6,1,0 %;      ! File version present
4716 0 literal FSCNS_FILESPEC = 1;             ! complete filespec
4717 0 literal FSCNS_NODE = 2;                  ! node:: field
4718 0 literal FSCNS_DEVICE = 3;                ! device: field
4719 0 literal FSCNS_ROOT = 4;                  ! [root.] field
4720 0 literal FSCNS_DIRECTORY = 5;             ! [directory] field
4721 0 literal FSCNS_NAME = 6;                  ! name field
4722 0 literal FSCNS_TYPE = 7;                  ! .typ field
4723 0 literal FSCNS_VERSION = 8;               ! ;version field
4724 0 literal FSCNS_ITEM_LEN = 8;
4725 0 literal FSCNS_ITEM_DEF = 8;
4726 0 macro FSCNSW_LENGTH = 0,0,16,0 %;        ! return length word
4727 0 macro FSCNSW_ITEM_CODE = 2,0,16,0 %;     ! item code value
4728 0 macro FSCNSL_ADDR = 4,0,32,0 %;          ! return length pointer
4729 0
4730 0 *** MODULE $RMEDEF ***
4731 0
4732 0         rms escape definitions
4733 0
4734 0         the following values identify various requests for non-standard rms
4735 0         functions.  they are currently input to the $modify function in the
4736 0         ctx field of the fab only if the esc bit is set in fop.  incorrect
4737 0         use of these capabilities could cause rms to fail, hence great caution
4738 0         should be exercised in their use.
4739 0
4740 0 literal RMESC_SETRFM = 1;                    ! change rfm, mrs, and fsz (if vfc) in ifab only
4741 0 literal RMESC_PPFECHO = 2;                  ! enable echo of SYSS$INPUT to SYSS$OUTPUT
4742 0
4743 0 *****
4744 0 *
4745 0 * Copyright (c) 1982, 1983
4746 0 * by DIGITAL Equipment Corporation, Maynard, Mass.
4747 0 *
4748 0 * This software is furnished under a license and may be used and copied
4749 0 * only in accordance with the terms of such license and with the
4750 0 * inclusion of the above copyright notice. This software or any other
4751 0 * copies thereof may not be provided or otherwise made available to any
4752 0 * other person. No title to and ownership of the software is hereby
4753 0 * transferred.
4754 0 *
4755 0 * The information in this software is subject to change without notice
4756 0 * and should not be construed as a commitment by DIGITAL Equipment
4757 0 * Corporation.
4758 0 *
4759 0 * DIGITAL assumes no responsibility for the use or reliability of its
4760 0 * software on equipment which is not supplied by DIGITAL.
4761 0 *
4762 0 *****
4763 0 *****
4764 0 Created 15-SEP-1984 22:55:15 by VAX-11 SDL V2.0      Source: 15-SEP-1984 22:50:36 _S255SDUA28:[RMS.SRC]RMSFILS
4765 0 *****
4766 0
4767 0
4768 0 *** MODULE $PLGDEF ***
4769 0 literal PLGSC_VER_NO = 1;                    ! current prolog version number
4770 0 literal PLGSC_VER_IDX = 2;                  ! new plg for indexed files
4771 0 literal PLGSC_VER_3 = 3;                    ! new plg for compression, space reclamation (plg 3)

```

```

4772 0 literal PLG$K_BLN = 120;
4773 0 literal PLG$C_BLN = 120;
4774 0 literal PLG$S_PLGDEF = 120;
4775 0 macro PLG$B_DBKTSIZ = 11,0,8,0 %; ! data bucket size
4776 0 macro PLG$R_FLAGS_OVERLAY = 16,0,8,0 %;
4777 0 macro PLG$B_FLAGS = 16,0,8,0 %; ! flag bits
4778 0 macro PLG$R_FLAGS_BITS = 16,0,8,0 %;
4779 0 macro PLG$V_NOEXTEND = 16,0,1,0 %; ! no extend allowed (rel)
4780 0 macro PLG$B_AVBN = 102,0,8,0 %; ! vbn of first area descriptor
4781 0 macro PLG$B_AMAX = 103,0,8,0 %; ! maximum number of areas
4782 0 macro PLG$W_DVBN = 104,0,16,0 %; ! first data bucket vbn
4783 0 macro PLG$L_MRN = 108,0,32,0 %; ! maximum record number (rel)
4784 0 macro PLG$L_EOF = 112,0,32,0 %; ! eof vbn (rel)
4785 0 macro PLG$W_VER_NO = 116,0,16,0 %; ! version number
4786 0 macro PLG$W_GBC = 118,0,16,0 %; ! default global buffer count
4787 0
4788 0 *** MODULE $DLCDEF ***
4789 0
4790 0
4791 0 relative file deletion control byte bit definitions
4792 0
4793 0 literal DLC$M_DELETED = 4;
4794 0 literal DLC$M_REC = 8;
4795 0 literal DLC$S_DLCDEF = 1;
4796 0 macro DLC$R_DLCDEF_BITS = 0,0,8,0 %;
4797 0 macro DLC$V_DELETED = 0,2,1,0 %; ! record deleted
4798 0 macro DLC$V_REC = 0,3,1,0 %; ! record exists (but may have been deleted)
4799 0
4800 0 *** MODULE $BKDEF ***
4801 0
4802 0 index bucket definition
4803 0
4804 0 this is the bucket format for RMS-11/RMS-32 index files.
4805 0
4806 0 literal BKT$K_OVERHDSZ = 14; ! length of bucket overhead
4807 0 literal BKT$C_OVERHDSZ = 14; ! length of bucket overhead
4808 0 literal BKT$M_LASTBKT = 1;
4809 0 literal BKT$M_ROOTBKT = 2;
4810 0 literal BKT$M_PTR_SZ = 24;
4811 0 literal BKT$C_ENDOVHD = 4; ! end of bucket overhead
4812 0 literal BKT$C_DATBKTOVH = 2; ! end of bucket overhead for data buckets
4813 0 literal BKT$C_DUPBKTOVH = 4; ! additional end of data bucket overhead
4814 0 ! when duplicates are allowed (LCB pointer)
4815 0 literal BKT$C_MAXBKTSIZ = 63; ! maximum bucket size
4816 0 literal BKT$S_BKDEF = 14;
4817 0 macro BKT$B_CHECKCHAR = 0,0,8,0 %; ! bucket check character
4818 0 macro BKT$R_AREANO_OVERLAY = 1,0,8,0 %;
4819 0 macro BKT$B_AREANO = 1,0,8,0 %; ! area number form which bucket was allocated
4820 0 macro BKT$B_INDEXNO = 1,0,8,0 %; ! index to which this bucket belongs (plg 3)
4821 0 macro BKT$W_ADRSAMPLE = 2,0,16,0 %; ! address sample - low 16 bits of first vbn in bucket
4822 0 macro BKT$R_FREESPACE_OVERLAY = 4,0,16,0 %;
4823 0 macro BKT$W_FREESPACE = 4,0,16,0 %; ! displacement in bucket of first free byte
4824 0 macro BKT$W_KEYFRESPACE = 4,0,16,0 %; ! pointer to key's free space (plg 3)
4825 0 macro BKT$R_NXTRECID_OVERLAY = 6,0,16,0 %;
4826 0 macro BKT$W_NXTRECID = 6,0,16,0 %; ! next available word record id (plg 3)
4827 0 macro BKT$R_NXTRECID_FIELDS = 6,0,16,0 %;
4828 0 macro BKT$B_NXTRECID = 6,0,8,0 %; ! next available record id

```

```

4829 0 macro BKT$B_LSTRECID = 7,0,8,0 %; ! last id in range
4830 0 macro BKT$B_NXTBKT = 8,0,32,0 %; ! vbn of next bucket
4831 0 macro BKT$B_LEVEL = 12,0,8,0 %; ! bucket level number
4832 0 macro BKT$B_BKTCB_OVERLAY = 13,0,8,0 %;
4833 0 macro BKT$B_BKTCB = 13,0,8,0 %; ! bucket control bits
4834 0 macro BKT$B_BKTCB_BITS = 13,0,8,0 %;
4835 0 macro BKT$V_LASTBKT = 13,0,1,0 %; ! last bucket in horizontal chain
4836 0 macro BKT$V_ROOTBKT = 13,1,1,0 %; ! root bucket
4837 0 macro BKT$V_PTR_SZ = 13,3,2,0 %;
4838 0 literal BKT$S_PTR_SZ = 2; ! size of vbn pointers in this bucket
4839 0
4840 0 *** MODULE $IRCDEF ***
4841 0
4842 0 index record definition
4843 0
4844 0 this is the definition of RMS-11/RMS-32 index file record formats
4845 0
4846 0 literal IRC$M_PTRSZ = 3;
4847 0 literal IRC$M_RECORDCB = 252;
4848 0 literal IRC$M_DELETED = 4;
4849 0 literal IRC$M_NOPTRSZ = 16;
4850 0 literal IRC$M_FIRST_KEY = 128;
4851 0 literal IRC$M_RRV = 8;
4852 0 literal IRC$M_NODUPCNT = 16;
4853 0 literal IRC$M_RU_DELETE = 32;
4854 0 literal IRC$M_RU_UPDATE = 64;
4855 0 literal IRC$C_IDXPTRBAS = 2; ! used to determine size of pointer in index
4856 0 literal IRC$C_IDXOVHDSZ = 1; ! includes record control byte
4857 0
4858 0 data bucket record
4859 0
4860 0 literal IRC$S_IRCDEF = 1;
4861 0 macro IRC$B_CONTROL = 0,0,8,0 %; ! record control byte
4862 0 macro IRC$R_CONTROL_BITS0 = 0,0,8,0 %;
4863 0 macro IRC$V_PTRSZ = 0,0,2,0 %;
4864 0 literal IRC$S_PTRSZ = 2; ! size of pointer
4865 0 macro IRC$V_RECORDCB = 0,2,6,0 %;
4866 0 literal IRC$S_RECORDCB = 6; ! record control bits
4867 0
4868 0 record control bits used only in primary data record and SDR array element
4869 0 control bytes
4870 0
4871 0 macro IRC$R_CONTROL_BITS1 = 0,0,8,0 %;
4872 0 macro IRC$V_DELETED = 0,2,1,0 %; ! record is deleted
4873 0 macro IRC$V_NOPTRSZ = 0,4,1,0 %; ! no RRV
4874 0 macro IRC$V_FIRST_KEY = 0,7,1,0 %;
4875 0
4876 0 record control bits used only in primary data record control bytes
4877 0
4878 0 macro IRC$R_CONTROL_BITS2 = 0,0,8,0 %;
4879 0 macro IRC$V_RRV = 0,3,1,0 %; ! rrv record
4880 0
4881 0 record control bits used only in prologue 2 SDR record control bytes
4882 0
4883 0 macro IRC$R_CONTROL_BITS3 = 0,0,8,0 %;
4884 0 macro IRC$V_NODUPCNT = 0,4,1,0 %; ! DUP_CNT field absent
4885 0

```

```

4886 0 | record control bits used only in prologue 3 RRV, UDR and SIDR record control
4887 0 | bytes of RU journalled files. (RU_UPDATE is set only in UDR record control
4888 0 | bytes)
4889 0
4890 0 macro IRC$R_CONTROL_BITS4 = 0,0,8,0 %;
4891 0 macro IRC$V_RU_DELETE = 0,5,1,0 %; | record is RU deleted
4892 0 macro IRC$V_RU_UPDATE = 0,6,1,0 %; | record is RU updated
4893 0
4894 0 | record control bits reserved for RMS-11 use only (these may not be re-defined
4895 0 | except for prologue 3 records)
4896 0
4897 0 | Bit number 5
4898 0 | Bit number 6
4899 0
4900 0
4901 0 | index bucket record
4902 0
4903 0 macro IRC$R_CONTROL_FIELDS4 = 0,0,8,0 %;
4904 0 macro IRC$T_BUCKETPTR = 1,0,0,0 %; | bucket pointer (not referenced in the code,
4905 0 | just present for consistency)
4906 0 literal IRC$S_IRCDEF1 = 3;
4907 0 macro IRC$B_ID = 1,0,8,0 %; | record id
4908 0 macro IRC$B_RRV_ID = 2,0,8,0 %; | rrv's id -- always in the same place
4909 0
4910 0 | prologue 3 data bucket record
4911 0
4912 0 literal IRC$C_DATSZFLD = 2; | size of size field in variable length records
4913 0 literal IRC$C_DATPTRBAS = 3; | used to determine size of RRV in data buckets
4914 0 literal IRC$C_DCNTSZFLD = 4; | size of duplicate count field in Plg 2 SIDRs
4915 0 literal IRC$C_DATOVHDSZ = 2; | includes the record control byte, and the id
4916 0 literal IRC$C_FIXOVHDSZ = 7; | the record overhead for fixed record
4917 0 literal IRC$C_VAROVHDSZ = 9; | record overhead for variable records
4918 0 literal IRC$C_RRVOVHDSZ = 7; | size of RRV
4919 0
4920 0 | prologue 3 constants
4921 0
4922 0 literal IRC$C_DATPTRBS3 = 4; | used to determine size of RRV in data buckets
4923 0 literal IRC$C_DATOVHSZ3 = 3; | record control byte, and id
4924 0 literal IRC$C_FIXOVHSZ3 = 9; | record overhead for fixed length records
4925 0 literal IRC$C_VAROVHSZ3 = 11; | record overhead for variable length records
4926 0 literal IRC$C_RRVOVHSZ3 = 9; | size of RRV
4927 0 literal IRC$C_SDROVHSZ3 = 2; | record overhead for SIDRs
4928 0 literal IRC$C_KEYCMPOVH = 2; | key compression overhead
4929 0 literal IRC$C_DATCMPOVH = 3; | data compression overhead
4930 0 literal IRC$S_IRCDEF2 = 5;
4931 0 macro IRC$W_ID = 1,0,16,0 %; | record id
4932 0 macro IRC$W_RRV_ID = 3,0,16,0 %; | rrv's id -- always in the same place
4933 0
4934 0 | constants
4935 0
4936 0
4937 0 | *** MODULE $KEYDEF ***
4938 0
4939 0 | definitions for the key descriptors in the prologue
4940 0
4941 0 | these definitions are associated w/ the plg and area definitions
4942 0

```

```

4943 0 Literal KEYSM-DUPKEYS = 1;
4944 0 Literal KEYSM-CHGKEYS = 2;
4945 0 Literal KEYSM-NULKEYS = 4;
4946 0 Literal KEYSM-IDX COMPR = 8;
4947 0 Literal KEYSM-INITIDX = 16;
4948 0 Literal KEYSM-KEY COMPR = 64;
4949 0 Literal KEYSM-REC COMPR = 128;
4950 0 Literal KEYS-KEYSC-MAX-DAT = 10;
4951 0 record
4952 0 Literal KEYS-KEYSC-MAX-PRIMARY = 6;
4953 0 primary key
4954 0 Literal KEYS-KEYSC-MAX-INDEX = 6;
4955 0 index and SDR key
4956 0 Literal KEYS-KEYSC-STRING = 0;
4957 0 Literal KEYS-KEYSC-SGNWORD = 1;
4958 0 Literal KEYS-KEYSC-UNSGNWORD = 2;
4959 0 Literal KEYS-KEYSC-SGNLONG = 3;
4960 0 Literal KEYS-KEYSC-UNSGNLONG = 4;
4961 0 Literal KEYS-KEYSC-PACKED = 5;
4962 0 Literal KEYS-KEYSC-SGNQUAD = 6;
4963 0 Literal KEYS-KEYSC-UNSGNQUAD = 7;
4964 0 Literal KEYS-KEYSC-MAX-DATA = 7;
4965 0 Literal KEYS-KEYSK-BLN = 96;
4966 0 Literal KEYS-KEYSC-BLN = 96;
4967 0 Literal KEYS-KEYSC-SPARE = 6;
4968 0 Literal KEYS-KEYSS-KEYDEF = 96;
4969 0 macro KEYS-KEYSL-IDXFL = 0,0,32,0 %;
4970 0 macro KEYS-KEYSW-NOFF = 4,0,16,0 %;
4971 0 macro KEYS-KEYSB-IANUM = 6,0,8,0 %;
4972 0 macro KEYS-KEYSB-LANUM = 7,0,8,0 %;
4973 0 macro KEYS-KEYSB-DANUM = 8,0,8,0 %;
4974 0 macro KEYS-KEYSB-ROOTLEV = 9,0,8,0 %;
4975 0 macro KEYS-KEYSB-IDXBKTSZ = 10,0,8,0 %;
4976 0 macro KEYS-KEYSB-DATBKTSZ = 11,0,8,0 %;
4977 0 macro KEYS-KEYSL-ROOTVBN = 12,0,32,0 %;
4978 0 macro KEYS-KEYSR-FLAGS-OVERLAY = 16,0,8,0 %;
4979 0 macro KEYS-KEYSB-FLAGS = 16,0,8,0 %;
4980 0 macro KEYS-KEYSR-FLAGS-BITS0 = 16,0,8,0 %;
4981 0 macro KEYS-KEYSV-DUPKEYS = 16,0,1,0 %;
4982 0 macro KEYS-KEYSV-CHGKEYS = 16,1,1,0 %;
4983 0 macro KEYS-KEYSV-NULKEYS = 16,2,1,0 %;
4984 0 macro KEYS-KEYSV-IDX COMPR = 16,3,1,0 %;
4985 0 macro KEYS-KEYSV-INITIDX = 16,4,1,0 %;
4986 0 macro KEYS-KEYSV-KEY COMPR = 16,6,1,0 %;
4987 0 macro KEYS-KEYSR-FLAGS-BITS1 = 16,0,8,0 %;
4988 0 macro KEYS-KEYSV-REC COMPR = 16,7,1,0 %;
4989 0 macro KEYS-KEYSB-DATATYPE = 17,0,8,0 %;
4990 0 macro KEYS-KEYSB-SEGMENTS = 18,0,8,0 %;
4991 0 macro KEYS-KEYSB-NULLCHAR = 19,0,8,0 %;
4992 0 macro KEYS-KEYSB-KEYSZ = 20,0,8,0 %;
4993 0 macro KEYS-KEYSB-KEYREF = 21,0,8,0 %;
4994 0 macro KEYS-KEYSW-MINRECSZ = 22,0,16,0 %;
4995 0 macro KEYS-KEYSW-IDXFILL = 24,0,16,0 %;
4996 0 macro KEYS-KEYSW-DATFILL = 26,0,16,0 %;
4997 0 macro KEYS-KEYSR-POSITION-OVERLAY = 28,0,16,0 %;
4998 0 macro KEYS-KEYSW-POSITION = 28,0,16,0 %;
4999 0 macro KEYS-KEYSW-POSITION0 = 28,0,16,0 %;

```

```

! (PLG3) Maximum size of a non-compressed data
! (PLG3) Maximum size of a non-compressed
! (PLG3) Maximum size of a non-compressed
string data type
signed binary word
unsigned binary word
signed binary long word
unsigned binary long word
packed decimal
signed binary quadword
unsigned binary quadword
maximum data type value allowed
length of key descriptor in the prologue (plg 3)
length of key descriptor in the prologue (plg 3)
these are spare words in key block (plg 3)
vbn for next key descriptor
offset to next key descriptor
index area number
level 1 area number
data area number
root level
index bucket size
data bucket size
root bucket pointer
flag bits
duplicate key values allowed
key value may change on $update operation
null key character enabled
index is compressed
index must be initialized
(PLG3) key is compressed in data record
(PLG3) Data record is compressed
data type for key
number of segments in key
"null" character
total key size
key of reference
minimum record length
index fill quantity
data fill quantity
key seg position
another name for position 0

```

```

5000 0 macro KEYSW_POSITION1 = 30,0,16,0 %; : position 1
5001 0 macro KEYSW_POSITION2 = 32,0,16,0 %; : position 2
5002 0 macro KEYSW_POSITION3 = 34,0,16,0 %; : position 3
5003 0 macro KEYSW_POSITION4 = 36,0,16,0 %; : position 4
5004 0 macro KEYSW_POSITION5 = 38,0,16,0 %;
5005 0 macro KEYSW_POSITION6 = 40,0,16,0 %;
5006 0 macro KEYSW_POSITION7 = 42,0,16,0 %;
5007 0 macro KEYSR_SIZE_OVERLAY = 44,0,8,0 %;
5008 0 macro KEYSB_SIZE = 44,0,8,0 %; : key segment size
5009 0 macro KEYSB_SIZE0 = 44,0,8,0 %; : another name for size
5010 0 macro KEYSB_SIZE1 = 45,0,8,0 %; : size 1
5011 0 macro KEYSB_SIZE2 = 46,0,8,0 %;
5012 0 macro KEYSB_SIZE3 = 47,0,8,0 %;
5013 0 macro KEYSB_SIZE4 = 48,0,8,0 %;
5014 0 macro KEYSB_SIZE5 = 49,0,8,0 %;
5015 0 macro KEYSB_SIZE6 = 50,0,8,0 %;
5016 0 macro KEYSB_SIZE7 = 51,0,8,0 %;
5017 0 macro KEYS KEYNAM = 52,0,0,0 %;
5018 0 literal KEYS KEYNAM = 32; : key name
5019 0 macro KEYSB_LDVBK = 84,0,32,0 %; : first data bucket
5020 0 macro KEYSR_TYPE_OVERLAY = 88,0,8,0 %;
5021 0 macro KEYSB_TYPE = 88,0,8,0 %; : key segment datatype (plg 3)
5022 0 macro KEYSB_TYPE0 = 88,0,8,0 %; : another name for first datatype (plg 3)
5023 0 macro KEYSB_TYPE1 = 89,0,8,0 %; : (plg 3)
5024 0 macro KEYSB_TYPE2 = 90,0,8,0 %; : (plg 3)
5025 0 macro KEYSB_TYPE3 = 91,0,8,0 %; : (plg 3)
5026 0 macro KEYSB_TYPE4 = 92,0,8,0 %; : (plg 3)
5027 0 macro KEYSB_TYPE5 = 93,0,8,0 %; : (plg 3)
5028 0 macro KEYSB_TYPE6 = 94,0,8,0 %; : (plg 3)
5029 0 macro KEYSB_TYPE7 = 95,0,8,0 %; : (plg 3)
5030 0
5031 0 :*** MODULE $AREADEF ***
5032 0
5033 0 :
5034 0 : definitions for the area descriptor in the prologue
5035 0 :
5036 0 :
5037 0 literal AREASC_CYL = 1; : cylindred alignment
5038 0 literal AREASC_LBN = 2; : logical block alignment
5039 0 literal AREASC_VBN = 3; : virtual block alignment
5040 0 literal AREASC_RFI = 4; : allocate close to related file by fid
5041 0 literal AREASM_HARD = 1;
5042 0 literal AREASM_ONC = 2;
5043 0 literal AREASM_CBT = 32;
5044 0 literal AREASM_CTG = 128;
5045 0 literal AREASK_BLN = 64; : length of area descriptor in the prologue
5046 0 literal AREASC_BLN = 64; : length of area descriptor in the prologue
5047 0 literal AREASS_AREADEF = 64;
5048 0 macro AREASB_FLAGS = 1,0,8,0 %; : not currently used
5049 0 macro AREASB_AREID = 2,0,8,0 %; : area id
5050 0 macro AREASB_ARBKTSZ = 3,0,8,0 %; : bucket size for area
5051 0 macro AREASW_VOLUME = 4,0,16,0 %; : relative volume number
5052 0 macro AREASB_ALN = 6,0,8,0 %; : extend allocation alignment
5053 0 macro AREASR_AOP_OVERLAY = 7,0,8,0 %;
5054 0 macro AREASB_AOP = 7,0,8,0 %; : alignment options
5055 0 macro AREASR_AOP_BITS = 7,0,8,0 %;
5056 0 macro AREASV_HARD = 7,0,1,0 %; : absolute alignment or nothing

```

```

5057 0 macro AREASV_ONC = 7,1,1,0 %;      locate on cylinder
5058 0 macro AREASV_CBT = 7,5,1,0 %;      contiguous best try
5059 0 macro AREASV_CTG = 7,7,1,0 %;      contiguous
5060 0 macro AREASL_AVAIL = 8,0,32,0 %;     available (returned) buckets
5061 0 macro AREASL_CVBN = 12,0,32,0 %;    start vbn for current extent
5062 0 macro AREASL_CNBLK = 16,0,32,0 %;   number of blocks in current extent
5063 0 macro AREASL_USED = 20,0,32,0 %;    number of blocks used
5064 0 macro AREASL_NXTVBN = 24,0,32,0 %;  next vbn to use
5065 0 macro AREASL_NXT = 28,0,32,0 %;     start vbn for next extent
5066 0 macro AREASL_NXBLK = 32,0,32,0 %;   number of blocks in next extent
5067 0 macro AREASW_DEQ = 36,0,16,0 %;     default extend quantity
5068 0 macro AREASL_LOC = 40,0,32,0 %;     start lbn on volume
5069 0 macro AREASW_RFI = 44,0,0,0 %;
5070 0 literal AREASW_RFI = 6;              related file id
5071 0 macro AREASL_TOTAL_ALLOC = 50,0,32,0 %; total block allocation
5072 0 macro AREASL_CHECK = 62,0,16,0 %;   checksum
5073 0
5074 0 *** MODULE $RJRDEF ***
5075 0
5076 0
5077 0 definitions for the journaling records in RMS journals
5078 0 for performance reasons, the BKT and BLK forms
5079 0 should be an integral number of quadwords.
5080 0
5081 0
5082 0 literal RJRSC_VER1 = 1;                journal version 1
5083 0 literal RJRSC_VER2 = 2;                journal version 2
5084 0 literal RJRSC_MAXVER = 2;              version limit
5085 0 literal RJRSC_NOENT = 0;               null type
5086 0 literal RJRSC_MAPPING = 1;             mapping entry
5087 0 literal RJRSC_FILENAME = 1;            mapping entry synonym
5088 0 literal RJRSC_RECORD = 2;              record entry
5089 0 literal RJRSC_BLOCK = 3;               block IO entry (at, etc...)
5090 0 literal RJRSC_BUCKET = 4;              ISAM bucket
5091 0 literal RJRSC_EXTEND = 5;              extend (AT, AI)
5092 0 literal RJRSC_AT_RECORD = 6;           audit trail record
5093 0 literal RJRSC_MAXTYP = 6;              entry-type limit
5094 0 literal RJRSC_SEQ = 0;                  sequential file org
5095 0 literal RJRSC_REL = 1;                  relative file org
5096 0 literal RJRSC_IDX = 2;                  indexed file org
5097 0 literal RJRSC_HSH = 3;                  hashed file org
5098 0 literal RJRSC_MAXORG = 3;               org limit
5099 0 literal RJRSC_NOOP = 0;                 null operation
5100 0 literal RJRSC_BUCKET = 1;              bucket-level I/O
5101 0 literal RJRSC_CLOSE = 2;                close
5102 0 literal RJRSC_CONNECT = 3;              connect
5103 0 literal RJRSC_CREATE = 4;               create
5104 0 literal RJRSC_DELETE = 5;               delete
5105 0 literal RJRSC_DISCONNECT = 6;           disconnect
5106 0 literal RJRSC_DISPLA = 7;               display
5107 0 literal RJRSC_ENTER = 8;                enter
5108 0 literal RJRSC_ERASE = 9;                erase
5109 0 literal RJRSC_EXTEND = 10;              extend
5110 0 literal RJRSC_FIND = 11;                find
5111 0 literal RJRSC_FLUSH = 12;               flush
5112 0 literal RJRSC_FREE = 13;                free
5113 0 literal RJRSC_GET = 14;                 get

```


H 16
15-Sep-1984 22:56:00
15-Sep-1984 22:55:58

VAX-11 Bliss-32 V4.0-742
_S255SDUA28:[RMS.OBJ]RMS.R32;1

Page 95
(9)

```
5114 0 literal RJRS_MODIFY = 15;      ! modify
5115 0 literal RJRS_NXTVOL = 16;     ! next volume
5116 0 literal RJRS_OPEN = 17;      ! open
5117 0 literal RJRS_PARSE = 18;     ! parse
5118 0 literal RJRS_PUT = 19;       ! put
5119 0 literal RJRS_READ = 20;      ! block I/O read
5120 0 literal RJRS_RELEASE = 21;   ! release
5121 0 literal RJRS_REMOVE = 22;    ! remove
5122 0 literal RJRS_RENAME = 23;    ! rename
5123 0 literal RJRS_REWIND = 24;    ! rewind
5124 0 literal RJRS_SEARCH = 25;    ! search
5125 0 literal RJRS_SPACE = 26;     ! block I/O space
5126 0 literal RJRS_TRUNCATE = 27;  ! truncate
5127 0 literal RJRS_UPDATE = 28;    ! update
5128 0 literal RJRS_WAIT = 29;      ! wait
5129 0 literal RJRS_WRITE = 30;     ! block I/O write
5130 0 literal RJRS_TPT = 31;       ! truncate on PUT
5131 0 literal RJRS_MAXOPER = 31;   ! oper limit
5132 0 literal RJRS_NOJNL = 0;      ! null jnl type
5133 0 literal RJRS_RMS_AI = 1;     ! after-image journal
5134 0 literal RJRS_RMS_BI = 2;     ! before-image journal
5135 0 literal RJRS_RMS_RU = 3;     ! recovery unit
5136 0 literal RJRS_MAXJNL = 3;     ! jnl type limit
5137 0 literal RJRSK_HDRLEN = 56;    ! common header len
5138 0 literal RJRSK_HDRLEN = 56;    ! common header len
5139 0 literal RJRSK_C_FIBLEN = 64;  !
5140 0 literal RJRSK_C_FIBLEN = 64;  !
5141 0 literal RJRSK_RECATRLEN = 32; !
5142 0 literal RJRSK_RECATRLEN = 32; !
5143 0 literal RJRSK_FILNAMLEN = 452; !
5144 0 literal RJRSK_FILNAMLEN = 452; !
5145 0 literal RJRSK_RECLEN = 72;    ! record entry len
5146 0 literal RJRSK_RECLEN = 72;    ! record entry len
5147 0 literal RJRSK_BLKLEN = 68;    ! block i/o entry len
5148 0 literal RJRSK_BLKLEN = 68;    ! block i/o entry len
5149 0 literal RJRSK_EXTLEN = 122;   ! extend entry len
5150 0 literal RJRSK_EXTLEN = 122;   ! extend entry len
5151 0 literal RJRSK_BKTLEN = 68;    ! bucket entry len
5152 0 literal RJRSK_BKTLEN = 68;    ! bucket entry len
5153 0 literal RJRSK_AT_RECLEN = 76; !
5154 0 literal RJRSK_AT_RECLEN = 76; !
5155 0 literal RJRSK_BLN = 452;      ! length of RJR descriptor in the prologue
5156 0 literal RJRSK_BLN = 452;      ! length of RJR descriptor in the prologue
5157 0 literal RJRSS_RJRDEF = 452;   !
5158 0 macro RJRSR_FLAGS_OVERLAY = 0,0,16,0 %; ! control flags
5159 0 macro RJRSR_FLAGS = 0,0,16,0 %; !
5160 0 macro RJRSR_FLAGS_BITS = 0,0,16,0 %; !
5161 0 macro RJRSB_VERSION = 2,0,8,0 %; ! RMS journal version #
5162 0 macro RJRSB_ENTRY_TYPE = 3,0,8,0 %; ! journal entry type
5163 0 macro RJRSB_ORG = 4,0,8,0 %;   ! file organization
5164 0 macro RJRSB_OPER = 5,0,8,0 %;  ! RMS operation id
5165 0 macro RJRSB_JNL_TYPE = 6,0,8,0 %; ! journaling type
5166 0 macro RJRST_JNLID = 8,0,0,0 %; !
5167 0 literal RJRSS_JNLID = 8;        !
5168 0 macro RJRST_VOLNAM = 8,0,0,0 %; !
5169 0 literal RJRSS_VOLNAM = 12;     ! volume name
5170 0 macro RJRST_FID = 20,0,0,0 %;  !
```

```

5171 0 literal RJRSS FID = 6; ! file id
5172 0 macro RJRSS ID DATE = 28,0,0,0 %;
5173 0 literal RJRSS ID DATE = 8; ! time
5174 0 macro RJRSS AT_STS = 36,0,32,0 %; ! status of operation
5175 0 macro RJRSS AT_STV = 40,0,32,0 %; ! secondary status
5176 0 macro RJRSS AT_CTX = 44,0,32,0 %; ! user FAB/RAB CTX field
5177 0 macro RJRSS ENTRY OVERLAY = 56,0,0,0 %;
5178 0 macro RJRSS FILENAME ENTRY = 56,0,0,0 %;
5179 0 literal RJRSS FILENAME ENTRY = 396;
5180 0 macro RJRSS ATR_FLAGS_OVERLAY = 60,0,32,0 %;
5181 0 macro RJRSS ATR_FLAGS = 60,0,32,0 %;
5182 0 macro RJRSS ATR_FLAGS BITS = 60,0,8,0 %;
5183 0 macro RJRSS ATR_UCHAR = 60,0,1,0 %; ! UCHAR attribute present
5184 0 macro RJRSS ATR_PROT = 60,1,1,0 %; ! PROT attribute present
5185 0 macro RJRSS ATR_UIC = 60,2,1,0 %; ! UIC attribute present
5186 0 macro RJRSS ATR_REC = 60,3,1,0 %; ! RECORD attributes present
5187 0 macro RJRSS ATR_EXPIRE = 60,4,1,0 %; ! EXPIRATION present
5188 0 macro RJRSS UIC = 64,0,32,0 %; ! owner UIC
5189 0 macro RJRSS PROT = 68,0,32,0 %; ! prot mask
5190 0 macro RJRSS ALLOC = 72,0,32,0 %; ! initial allocation (audit)
5191 0 macro RJRSS UCHAR = 76,0,32,0 %; ! user characteristics (create)
5192 0 macro RJRSS EXPIRE = 80,0,0,0 %;
5193 0 literal RJRSS EXPIRE = 8; ! expiration date (create)
5194 0 macro RJRSS FNS = 88,0,8,0 %; ! size of file name
5195 0 macro RJRSS FAC = 90,0,8,0 %; ! file access (audit)
5196 0 macro RJRSS SHR = 91,0,8,0 %; ! sharing allowed (audit)
5197 0 macro RJRSS DID = 92,0,0,0 %;
5198 0 literal RJRSS DID = 6; ! directory ID (create, volume recovery)
5199 0 macro RJRSS C_FIB = 100,0,0,0 %;
5200 0 literal RJRSS C_FIB = 64; ! FIB (create)
5201 0 macro RJRSS REC_ATTR = 164,0,0,0 %;
5202 0 literal RJRSS REC_ATTR = 32; ! record attributes (create)
5203 0 macro RJRSS FILENAME = 196,0,0,0 %;
5204 0 literal RJRSS FILENAME = 256; ! full filename
5205 0 macro RJRSS RECORD ENTRY = 56,0,0,0 %;
5206 0 literal RJRSS RECORD ENTRY = 16; ! record entry
5207 0 macro RJRSS CHKSUM = 60,0,32,0 %; ! checksum of old record
5208 0 macro RJRSS RFA_OVERLAY = 64,0,0,0 %;
5209 0 macro RJRSS RFA = 64,0,0,0 %;
5210 0 literal RJRSS RFA = 6; ! RFA of record
5211 0 macro RJRSS RFA_FIELDS = 64,0,0,0 %;
5212 0 macro RJRSS RFA0 = 64,0,32,0 %; ! alternate RFA def
5213 0 macro RJRSS RFA4 = 68,0,16,0 %;
5214 0 macro RJRSS RRN = 64,0,32,0 %; ! relative record number
5215 0 macro RJRSS RSIZE = 70,0,16,0 %; ! record size
5216 0 macro RJRSS RIMAGE = 72,0,0,0 %; ! record date
5217 0
5218 0 ! The block entry is common to both AT and block IO journaling.
5219 0
5220 0 macro RJRSS BLOCK ENTRY = 56,0,0,0 %;
5221 0 literal RJRSS BLOCK ENTRY = 12; ! block entry
5222 0 macro RJRSS BLOCK_VBN = 60,0,32,0 %; ! vbn of block
5223 0 macro RJRSS BLOCK_SIZE = 64,0,32,0 %; ! transfer size (AT)
5224 0 macro RJRSS BLOCK = 68,0,0,0 %; ! block data
5225 0
5226 0 ! The extend entry is common to both AT and AI journaling.
5227 0

```

```

5228 0 macro RJR$R_EXTEND ENTRY = 56,0,0,0 %;
5229 0 literal RJR$$ EXTEND ENTRY = 66; ! extend entry
5230 0 macro RJR$T_FILL_EXT2 = 60,0,0,0 %;
5231 0 literal RJR$$ FILL_EXT2 = 32; ! no longer FIB
5232 0 ! currently unused
5233 0 macro RJR$R_EXT_FLAGS_OVERLAY = 92,0,32,0 %;
5234 0 macro RJR$R_EXT_FLAGS = 92,0,32,0 %;
5235 0 macro RJR$R_EXT_FLAGS_BITS = 92,0,8,0 %;
5236 0 macro RJR$V_EXT_USE_XAB = 92,0,1,0 %; ! ALL XAB fields present
5237 0
5238 0 Fields EXT_AOP (unused) through EXT_RFI are in same relative locations as
5239 0 the same fields in allocation XAB.
5240 0
5241 0 macro RJR$B_EXT_AOP = 100,0,8,0 %; ! align options
5242 0 macro RJR$B_EXT_ALN = 101,0,8,0 %; ! alignment boundary
5243 0 macro RJR$W_EXT_VOL = 102,0,16,0 %; ! relative volume number
5244 0 macro RJR$R_EXT_LOC = 104,0,32,0 %; ! location
5245 0 macro RJR$R_EXT_ALQ = 108,0,32,0 %; ! allocation quantity
5246 0 macro RJR$W_EXT_DEQ = 112,0,16,0 %; ! default extension
5247 0 macro RJR$B_EXT_BKZ = 114,0,8,0 %; ! bucket size
5248 0 macro RJR$B_EXT_AID = 115,0,8,0 %; ! area ID
5249 0 macro RJR$W_EXT_RFI = 116,0,0,0 %;
5250 0 literal RJR$$ EXT_RFI = 6; ! related file IFI
5251 0 macro RJR$T_EXT_ENDALL = 122,0,0,0 %; ! end of all info
5252 0 macro RJR$R_BUCKET ENTRY = 56,0,0,0 %;
5253 0 literal RJR$$ BUCKET ENTRY = 12; ! BUCKET entry
5254 0 macro RJR$R_BRT_VBN = 60,0,32,0 %; ! bucket vbn
5255 0 macro RJR$W_BKT_SIZE = 64,0,16,0 %; ! bucket size
5256 0 macro RJR$W_JBKT_SIZE = 66,0,16,0 %; ! actual size of
5257 0 ! journaled bucket data
5258 0 macro RJR$T_BUCKET = 68,0,0,0 %; ! bucket data
5259 0 macro RJR$R_AT_RECORD = 56,0,0,0 %;
5260 0 literal RJR$$ AT_RECORD = 20;
5261 0 macro RJR$R_AT_RDP = 60,0,32,0 %; ! record options
5262 0 macro RJR$B_AT_KRF = 64,0,8,0 %; ! key of reference
5263 0 macro RJR$B_AT_KSZ = 65,0,8,0 %; ! key size
5264 0 macro RJR$B_AT_RAC = 66,0,8,0 %; ! record access mode
5265 0 macro RJR$R_AT_RFA_OVERLAY = 68,0,0,0 %;
5266 0 macro RJR$W_AT_RFA = 68,0,0,0 %;
5267 0 literal RJR$$ AT_RFA = 6; ! RFA of record
5268 0 macro RJR$R_AT_RFA_FIELDS = 68,0,0,0 %;
5269 0 macro RJR$R_AT_RFA0 = 68,0,32,0 %; ! alternate RFA def
5270 0 macro RJR$W_AT_RFA4 = 72,0,16,0 %;
5271 0 macro RJR$R_AT_RRN = 68,0,32,0 %; ! relative record number
5272 0 macro RJR$T_AT_KEY = 76,0,0,0 %; ! key if used
5273 0
5274 0 The FILENAME entry is used to describe filename information required to:
5275 0
5276 0 1. Identify a stream of journal entries with a particular file.
5277 0 Used in particular for AI volume recovery or roll back RUs on remount.
5278 0
5279 0 2. Record the information required to re-create a file for AI journaling.
5280 0
5281 0 3. Record all the required audit-trail information for a $CREATE.
5282 0

```

COMMAND QUALIFIERS

```
;
;      BLISS/LIB=LIB$:RMS.L32/LIS=LISS$:RMS.LST LIB$:RMS.R32
; Run Time:      00:37.5
; Elapsed Time:  00:42.1
; Lines/CPU Min: 8451
; Lexemes/CPU-Min: 76835
; Memory Used:  590 pages
; Library Precompilation Complete
```


0314 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY